UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE CIENCIAS FISICO MATEMATICAS

SISTEMAS OPERATIVOS

T E S I S

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN CIENCIAS COMPUTACIONALES

P R E S E N T A

MA. MAGDALENA BARRERA RODRIGUEZ

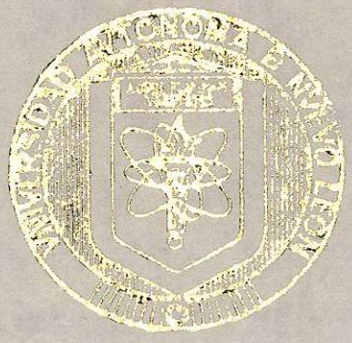
CD. UNIVERSITARIA, SAN NICOLAS DE LOS GARZA,

DICIEMBRE DE 1988


1080171496

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE CIENCIAS FISICO MATEMATICAS



SISTEMAS OPERATIVOS

T E S I S

QUE PARA OBTENER EL TITULO DE
LICENCIADO EN CIENCIAS COMPUTACIONALES

P R E S E N T A

MA. MAGDALENA BARRERA RODRIGUEZ

CD. UNIVERSITARIA, SAN NICOLAS DE LOS GARZA,
DICIEMBRE DE 1988

DEDICO ESTA TESIS

A DIOS

QUIEN ME HA DADO LA SUFICIENTE FORTALEZA PARA VENCER CADA OBSTACULO QUE SE ME HA PRESENTADO EN LA VIDA.

A MIS PADRES

SR. JOSE ARTEMIO BARRERA LOPEZ Y
SRA. IMELDA ANA RODRIGUEZ DE BARRERA

Y A MI HERMANO

ING. ARTEMIO BARRERA RODRIGUEZ JR.

COMO UN TESTIMONIO DE GRATITUD Y ETERNO RECONOCIMIENTO AL APOYO MORAL QUE SIEMFRE ME HAN BRINDADO Y CON EL CUAL HE LOGRADO FINALIZAR MIS ESTUDIOS PROFESIONALES QUE REPRESENTAN PARA MI LA MEJOR DE LAS HERENCIAS.

GRACIAS POR EL APOYO Y EJEMPLO QUE EN CADA SEGUNDO DE MI VIDA ME HAN BRINDADO, FOR SUS CUIDADOS, AMOR, TERNURA, COMPRENSION Y DESVELOS, Y POR SUS SABIOS CONSEJOS QUE ME HAN ORIENTADO POR EL CAMINO RECTO DE LA VIDA.

PERO GRACIAS PRINCIPALMENTE POR TENER EN USTEDES A MIS MEJORES AMIGOS.

A USTEDES POR TODO ELLO MI ETERNO Y SINCERO AGRADECIMIENTO YA QUE LOS AMO CON TODO MI CORAZON.

A MIS MAESTROS

LOS CUALES ME DIERON PARTE DE SUS CONOCIMIENTOS Y POR SIEMPRE ESTARE AGRADECIDA DE ELLOS.

A MI ASESOR

LIC. OSCAR DE JESUS AGUILAR DE LA ROSA
QUIEN ME HA DADO SU GRAN AYJDA PARA LA ELABORACION DE ESTA TESIS.

CONTENIDO

1. INTRODUCCION

2. HARDWARE, SOFTWARE, FIRMWARE

INTRODUCCION

1.1 OBSERVACIONES

POCA GENTE RECUERDA EL TIEMPO EN EL QUE NO HABIA SISTEMAS OPERATIVOS. LOS SISTEMAS OPERATIVOS SON USADOS EN COMPUTADORAS QUE VAN DESDE LOS MAINFRAMES GIGANTES A LAS PC S. EL SISTEMA OPERATIVO REPRESENTA PARA EL USUARIO EL PERFIL DE LA COMPUTADORA MAS QUE EL HARDWARE MISMO: ESTO OCURRE POR EJEMPLO, CON CP/M DONDE ESTE SE HA VUELTO TAN STANDARD, QUE MUCHAS MANUFACTURERAS CONSTRUYEN HARDWARE PARA SOPORTAR CP/M. ES DECIR CP/M PROPORCIONA LA PARTE FUNCIONAL.

EN 1960, SE PODRIA HABER DEFINIDO S.O. COMO "EL SOFTWARE QUE CONTROLA EL HARDWARE". HOY MUCHAS FUNCIONES TIENDEN A EMIGRAR DE SOFTWARE A FIRMWARE (MICROCODIGO), LAS CUALES PUEDEN PRONTO EXCEDER A LAS QUE ESTAN EN SOFTWARE.

ES NECESARIO UNA MEJOR DEFINICION: VEMOS UN S.O. COMO LOS PROGRAMAS IMPLANTADOS EN SOFTWARE O FIRMWARE, QUE HACEN UTILIZABLE EL HARDWARE.

EL HARDWARE PROVEE EL PODER DE COMPUTO, Y EL S.O HACE ESTE PODER DE COMPUTO CONVENIENTEMENTE USABLE. TAMBIEN MANEJA EL HARDWARE DE TAL MANERA QUE SE OBTENGA UN BUEN PERFORMANCE (EJECUCION). COMO QUIERA QUE SEA QUE LO DEFINAMOS: LOS S.O. SON UNA PARTE INTEGRAL DEL MEDIO AMBIENTE COMPUTACIONAL , DEBEN SER ENTENDIDOS EN ALGUN GRADO POR TODO USUARIO COMPUTACIONAL.

LOS S.O. DISPONIBLES CONSUMEN UNA GRAN PARTE DE LOS RECURSOS COMPUTACIONALES PARA SU FUNCIONAMIENTO, LO CUAL NO ES BIEN VISTO POR LOS USUARIOS.

ACTUALMENTE LOS VENDEDORES PROPORCIONAN S.O. MAS SIMPLES QUE NOS DAN UN MEDIO AMBIENTE COMPUTACIONAL MAS SIMPLE , CONVENIENTE, QUE SATISFACE LAS NECESIDADES ESPECIFICAS DE USUARIOS INDIVIDUALES.

UN S.O. ES PRIMERAMENTE UN MANEJADOR DE RECURSOS , EL RECURSO PRIMARIO QUE MANEJA ES EL HARDWARE. PROPORCIONA MUCHAS FACILIDADES INCLUYENDO:

* DEFINICION DE LA INTERFASE CON EL USUARIO
* COMPARTIR RECURSOS ENTRE USUARIOS
* PERMITE A LOS USUARIOS COMPARTIR DATOS ENTRE ELLOS MISMOS
* ASIGNAR RECURSOS A LOS USUARIOS
* FACILITA I/O
* RECUPERA ERRORES

LOS RECURSOS CLAVE QUE UN S.O. MANEJA SON:

* PROCESADOR(ES)
* ALMACENAMIENTO
* DISPOSITIVOS DE I/O
* DATOS

EL S.O. HACE INTERFASE CON:

* OPERADORES COMPUTACIONALES
* PROGRAMADORES DE APLICACIONES

* PROGRAMADORES DE SISTEMAS (SYSTEM S PROGRAMMER S)
* PERSONAL ADMINISTRATIVO
* PROGRAMAS
* HARDWARE
* USUARIOS

LOS USUARIOS SON LOS CLIENTES EN UN MEDIO AMBIENTE COMPUTACIONAL QUE UTILIZAN LA COMPUTADORA PARA EFECTUAR ALGUN TRABAJO UTIL.
LOS OPERADORES DE COMPUTADORA ES GENTE QUE TIENE LA RESPONSABILIDAD DE MONITOREAR EL SISTEMA OPERATIVO RESPONDIENDO A SOLICITUDES PARA INTERVENCION, MONTANDO Y DESMONTANDO CINTAS , DISCOS, CARGANDO , DESCARGANDO TARJETAS, PONIENDO LAS FORMAS CORRECTAS EN LAS IMPRESORAS Y QUE ESTEN BIEN ALINEADAS, ETC; SON NECESARIOS EN LA OPERACION DEL SISTEMA, PUES HACEN LAS FUNCIONES QUE NO HAN SIDO AUTOMATIZADAS.

LOS PROGRAMADORES DE SISTEMAS (SYSTEMS PROGRAMMERS) ESTAN RELACIONADOS CON EL MANTENIMIENTO DEL SISTEMA OPERATIVO, ADAPTANDOLO A LAS NECESIDADES DE LA INSTALACION Y MODIFICANDOLO PARA SOPORTAR NUEVOS DISPOSITIVOS.

LOS ADMINISTRADORES DE SISTEMAS ESTABLECEN LAS POLITICAS E INTERACTUAN CON EL S.O. PARA ASEGURAR QUE ESAS POLITICAS SON IMPLANTADAS EN FORMA APROPIADA.

LOS PROGRAMAS INTERACTUAN EL CON S.O. POR INSTRUCCIONES ESPECIALES TALES COMO: SUPERVISOR DE LLAMADAS (SUPERVISOR CALLS), MONITOR DE LLAMADAS (MONITOR CALLS), CUESTIONAMIENTOS EJECUTIVOS (EXECUTIVE REQUESTS) ETC. QUE PERMITEN AL USUARIO OBTENER SERVICIOS DEL S.O.

EL S.O. ES EL USUARIO MAS CONFIABLE (THE MOST TRUSTED USER).

1.2 GENERACIONES DE S.O.

LA SERIE DE CAMBIOS QUE HAN SUFRIDO LOS S.O. SE LES LLAMA GENERACIONES.

EN HARDWARE, AVENCES EN COMPONENTES:

1a.GENERACION (TUBOS AL VACIO)
2a.GENERACION (TRANSITORES)
3a.GENERACION (CIRCUITOS INTEGRADOS)
4a.GENERACION (GRAN ESCALA , VLSI)

CONSECUENCIAS:

* REDUCCION EN COSTOS
* REDUCCION EN TAMAN)
* REDUCCION EN EMISION DE CALOR
* INCREMENTO EN VELOCIDAD
* INCREMENTO EN CAPACIDAD DE ALMACENAMIENTO

1.3GENERACION ZERO (1940 S)

NO HABIA S.O. EL USUARIO TENIA EL ACCESO COMPLETO AL

LENGUAJE DE MAQUINA.

1.4 PRIMERA GENERACION (1950 S)

EL S.O. FUE DISEÑADO PARA SUAVIZAR LA TRANSICION ENTRE JOBS. ANTES, SE PERDIA MUCHO TIEMPO ENTRE LA TERMINACION DE UN JOB Y LA INICIACION DE OTRO. AQUI SE INICIA EL PROCESAMIENTO BATCH EN EL CUAL LOS JOBS SE AGRUPABAN EN PAQUETES. UNA VEZ QUE UN JOB ESTABA CORRIENDO TENIA EL CONTROL TOTAL DE LA MAQUINA AL TERMINAR (NORMAL O ANORMAL) EL CONTROL SE REGRESABA AL S.O. Y ESTE LEIA E INICIABA EL SIGUIENTE.

1.5 LA 2a. GENERACION (PRINCIPIOS DE LOS 1960 S)

SE CARACTERIZA POR EL DESARROLLO DE SISTEMAS COMPARTIDOS CON MULTIPROGRAMACION Y LOS PRINCIPIOS DE MULTIPROCESAMIENTO. EN SISTEMAS DE MULTIPROGRAMACION VARIOS PROGRAMAS ESTAN EN EL ALMACENAMIENTO PRINCIPAL A LA VEZ Y EL PROCESADOR ES SWITCHEADO RAPIDAMENTE ENTRE JOBS.
EN SISTEMAS DE MULTIPROCESAMIENTO VARIOS PROCESADORES SON USADOS EN UN SOLO SISTEMA COMPUTACIONAL PARA INCREMENTAR EL PODER DE COMPUTO DE LA MAQUINA.

APARECE LA INDEPENDENCIA DE DISPOSITIVOS:

SISTEMAS DE TIEMPO COMPARTIDO (TIMESHARING) FUERON DESARROLLADOS EN LOS CUALES LOS USUARIOS PODIAN INTERACTUAR DIRECTAMENTE CON LA COMPUTADORA A TRAVES DE TERMINALES (COMO MAQUINAS DE ESCRIBIR). LOS SISTEMAS DE TIEMPO COMPARTIDO (TIMESHARING) OPERAN EN UN MODO INTERACTIVO O CONVENCIONAL CON LOS USUARIOS. ESTO HIZO POSIBLE UN AVANCE EN EL PROCESO DE DESARROLLO DE PROGRAMAS, YA QUE LA LOCALIZACION Y CORRECCION DE ERRORES EN PROGRAMAS SE HIZO EN SEGUNDOS O MINUTOS, SUPERO EL RETARDO QUE SE TENIA EN EL PROCESAMIENTO BATCH DE HORAS O DIAS.

APARECEN LOS SISTEMAS DE TIEMPO REAL EN EL CUAL LAS COMPUTADORAS FUERON UTILIZADAS EN EL CONTROL DE PROCESOS INDUSTRIALES, POR EJEMPLO EN LAS REFINERIAS. SISTEMAS MILITARES DE TIEMPO REAL FUERON DESARROLLADOS PARA MONITOREAR MILES DE PUNTOS DE POSIBLES ATAQUES AL ENEMIGO. LOS SISTEMAS DE TIEMPO REAL SE CARACTERIZAN PORQUE PROVEEN RESPUESTA INMEDIATA.

1.6 LA TERCERA GENERACION (MITAD DE LOS 60 S A MITAD 70 S)

ESTA GENERACION INICIA EN 1964 CON LA INTRODUCCION DE LA FAMILIA DE LA IBM/360; LAS CUALES ERAN DE PROPOSITO GENERAL. ESTE CONCEPTO VENDIO MUCHOS EQUIPOS PERO TUVO UN PRECIO, LOS USUARIOS CON APLICACIONES PARTICULARES QUE NO REQUERIAN DE

LA POTENCIA OFRECIDA PASARON EN TIEMPO DE OVERHEAD, ENTRENAMIENTO, DEBUGGING, MANTENIMIENTO, ETC.

LOS SISTEMAS DE LA 3a. GENERACION FUERON (SISTEMAS MULTIMODALES) ALGUNOS DE ELLOS SOPORTARON SIMULTANEAMENTE BATCH, TIMESHARING, TIEMPO REAL Y MULTIPROCESAMIENTO. FUERON SISTEMAS GRANDES Y CAROS. NADA COMO ELLOS HABIA SIDO CONSTRUIDO ANTES Y MUCHOS DE ELLOS FUERA DEL PRESUPUESTO Y TIEMPO ASIGNADO. ESTOS SISTEMAS INTRODUCIERON UNA GRAN COMPLEJIDAD EN EL MEDIO AMBIENTE COMPUTACIONAL A LA QUE EL USUARIO NO ESTABA ACOSTUMBRADO. INTERPUCIERON UNA CAPA DE SOFTWARE ENTRE EL USUARIO Y EL HARDWARE , DE TAL MANERA QUE EL USUARIO TENIA SOLO LA VISTA DEL SOFTWARE PARA REALIZAR SUS TAREAS. LOS USUARIOS TUVIERON QUE FAMILIARIZARSE CON (JOB CONTROL LANGUAGES) COMPLEJOS. LOS S.O. DE ESTA GENERACION FUERON UN GRAN AVANCE, PERO UN CASTIGO PARA LOS USUARIOS.

1.7 CUARTA GENERACION (MITAD DE LOS 70 s AL PRESENTE)

SON EL ESTADO DEL ARTE EN S.O. MUCHOS DISEÑADORES Y USUARIOS SUFREN TODAVIA LA EXPERIENCIA DE LA 3a. GENERACION Y TIENEN MUCHO CUIDADO DE NO VERSE INVOLUCRADOS CON S.O. COMPLEJOS.

CARACTERISTICAS:

- AMPLIO USO DE REDES COMPUTACIONALES Y PROCESAMIENTO EN LINEA (ON-LINE)
- EL MICROPROCESADOR HA HECHO POSIBLE EL DESARROLLO DE LAS PC S
- DISMINUCION EN LOS COSTOS
- PC S EQUIPADAS CON INTERFASES DE COMUNICACION DE DATOS TAMBIEN SIRVEN COMO TERMINALES
- EL USUARIO SE PUEDE COMUNICAR CON SISTEMAS GRAFICAMENTE DISPERSOS
- INCREMENTO EN LOS PROBLEMAS DE SEGURIDAD
- ENCRIPCION HA RECIBIDO MUCHA ATENCION
- INCREMENTO EN LA RELACION CON REDES A LOS SISTEMAS COMPUTACIONALES
- SISTEMAS AMIGABLES (USER FRIENDLY)
- SISTEMAS MANEJADOS POR MENU (MENU-DRIVE)
- EL CONCEPTO DE MAQUINAS VIRTUALES ES AMPLIAMENTE USADO. EL USUARIO SE OLVIDA DE LOS DETALLES FISICOS DE LOS SITEMAS COMPUTACIONALES O REDES Y SOLO TIENE LA VISTA CREADA POR EL SISTEMA OPERATIVO (MAQUINA VIRTUAL)
- SISTEMAS DE BASE DE DATOS HAN TOMADO IMPORTANCIA CENTRAL.
- EL CONCEPTO DE PROCESAMIENTO DISTRIBUIDO DE DATOS SE HA ESTABLECIDO FIRMEMEMINIMIZAR EL TIEMPO OCIOSO ENTRE JOB Y JOB

- EMERGE EL CONCEPTO DE SISTEMAS DE NOMBRES DE ARCHIVOS, INICIANDO LA INDEFENDENCIA ENTRE DISPOSITIVOS
- A FINALES DE 1950, LOS VENDEDORES OFRECIAN LAS SIGUIENTES CARACTERISTICAS DE LOS S.O.
    * PROCESAMIENTO BATCH
    * RUTINAS STANDARD DE I/O
    * REDUCCION DEL OVERHEAD AL INICIAR UN JOB
    * TECNICAS MINIMIZAR EL

  TIEMPO OCIOSO ENTRE JOB Y JOB
- EMERGE EL CONCEPTO DE SISTEMAS DE NOMBRES DE ARCHIVOS, INICIANDO LA INDEPENDENCIA ENTRE DISPOSITIVOS
- A FINALES DE 1950, LOS VENDEDORES OFRECIAN LAS SIGUIENTES CARACTERISTICAS DE LOS S.O.
    * PROCESAMIENTO BATCH
    * RUTINAS STANDARD DE I/O
    * REDUCCION DEL OVERHEAD AL INICIAR UN JOB
    * TECNICAS DE RECUPERACION DE ERRORES
    * JCL'S

1.8 DESARROLLO EN LOS INICIOS DE LOS 60's

CARACTERISTICAS:
- LOS S.O. TENIAN FACILIDADES MUCHO MUY PODEROSAS
- COMPETIDORES: CDC, HONEYWELL I, NCR, RCA, BURROUGHS, GE, IBM, PHILCO, UNIVAC.
- OBJETIVOS: MEJORAR EL TRABAJO PROCESADO POR UNIDAD DE TIEMPO (THROUGHPUT)
- EMPLEO DE MULTIPROGRAMACION
- SISTEMAS DE MULTIPROCESAMIENTO
- SISTEMAS DE RESERVACIONES SABRE
- SISTEMAS CONVENCIONALES Y SISTEMAS EN LINEA (ON-LINE)
- ORGANIZACIONES DE ALMACENAMIENTO DE ACCESO DIRECTO
- S.O. CON CONTROL DE PROGRAMA PRINCIPAL (MASTER CONTROL PROGRAM)

1.9 FAMILIA DE COMPUTADORAS IBM/360 (1963)

- SURGEN ANTE EL PROBLEMA DE INCOMPATIBILIDAD
- UTILIZAN EL S.O. OS/360
- INCREMENTA EL PODER DE COMPUTO AL IR HACIA ARRIBA EN MODELOS
- SURGEN LOS SIMULADORES Y EMULADORES LOS CUALES HACEN UNA COMPUTADORA PARECER COMO SI FUERA OTRA

1.10 REACCION DE LA INDUSTRIA AL SISTEMA/360

LOS FABRICANTES DE COMPUTADORAS TUBIERON QUE DESARROLLAR ESTRATEGIAS PARA COMPETIR:
- COPIAR LA LECTURA DE LA 360 Y PROVEER UN S.O. SIMILAR

A OS/360 A BAJO COSTO FALLO.
- DESARROLLO DE PRODUCTOS NO COMPATIBLES CON LA 360, PERO MAS POTENTES Y DE MAS EFECTIVIDAD EN COSTO
- DESARROLLAR SISTEMAS MAS MEJORADOS QUE LAS LINEAS DE IBM 1400, 7000, ATRAER CLIENTES DE IBM, UNIVAC Y CDC.
- IBM TUVO QUE DESARROLLAR OTROS S.O. DE MAS FUNCIONES Y FACILIDADES ANTE LAS DEMANDAS DE LOS USUARIOS
  * DOS/360 PARA LAS 360 PEQUEÑAS
  * OS/MFT PARA LAS MEDIANAS Y GRANDES
  * OS/MVT PARA LAS GRANDES
  * CP-67 CMS PARA LA 360/67

1.11 SISTEMAS DE TIEMPO COMPARTIDO

A FINALES DE 1950 Y PRINCIPIOS DE 1960, LOS INVESTIGADORES DESARROLLARON VARIOS SISTEMAS DE TIEMPO COMPARTIDO Y SE PROBO EL VALOR DE ELLOS EN EL APOYO DE ACTIVIDADES DE DESARROLLO DE PROGRAMAS Y REVELO EL TREMENDO IMPACTO QUE EL COMPARTIR PROGRAMAS Y DATOS ENTRE USUARIOS TIENE SOBRE LA PRODUCTIVIDAD Y CREATIVIDAD EN LAS COMUNIDADES DE COMPUTACION. ESTOS SISTEMAS FUERON: CTSS, MULTICS, TSS PARA LA 360/67, CP-67 CMS, PCA/VMOS, KRONOS PARA LA CDC-6600

1.12 INGENIERIA DE SOFTWARE

EN EL DESARROLLO DE LOS S.O. SE CONSTRUYERON MUCHOS PROGRAMAS Y SE VIO QUE ERA DIFICIL EL DEPURARLOS, CORREGIRLOS, MANTENERLOS, ETC. DEBIDO A QUE NO SE HABIA SEGUIDO UNA TECNICA DE TAL MANERA QUE SE INVERTIA MUCHO TIEMPO Y COSTO. ESTO MISMO OCURRIA EN EL DESARROLLO DE OTROS SISTEMAS. ASI SURGE LA NECESIDAD DE ESTABLECER EL CAMPO DE INGENIERIA DE SOFTWARE

1.13 SEPARACION DE SOFTWARE Y HARDWARE

HASTA INICIAR 1970 LOS VENDEDORES OFRECIAN HARDWARE Y SOFTWARE SIN CARGO Y ELUDIAN LA RESPONSABILIDAD POR SU CALIDAD.

IBM EMPEZO A FACTURAR SEPARADAMENTE EL HARDWARE Y EL SOFTWARE AUNQUE CONTINUO DANDO ALGUN SOFTWARE SIN CARGO.

CONSECUENCIAS:
- MAS RESPONSABILIDAD DEL VENDEDOR PORQUE EL SOFTWARE FUNCIONE CORRECTAMENTE
- CREACION DE INDUSTRIA DE SOFTWARE INDEPENDIENTE
- RESPUESTA SIMILAR DE OTROS FABRICANTES DE COMPUTADORAS
- INCENTIVO PARA EL USUARIO
- MODULARIDAD EN EL SOFTWARE

1.14 TENDENCIAS

- ABATIMIENTO EN COSTOS Y TAMANO FISICO
- INCREMENTO EN VELOCIDAD Y CAPACIDAD DE ALMACENAMIENTO
- CONTINUACION DE LA ESCALA DE INTEGRACION (VLSI A ULSI)
- MAS COMUN EL USO DE MULTIPROCESAMIENTO
- MIGRACION A MICROCODIGO DE MUCHAS FUNCIONES
- CONTROL DISTRIBUIDO
- EJECUCION EFICIENTE DE PROGRAMAS CONCURRENTES
- PARALELISMO
- MAQUINAS VIRTUALES
- FAMILIAS DE COMPUTADORAS

HARDWARE, SOFTWARE, FIRMWARE

2.1 INTRODUCCION

HARDWARE: CONSISTE DE LOS DISPOSITIVOS DE UN SISTEMA COMPUTACIONAL, SUS PROCESADORES, SU ALMACENAMIENTO I/O Y COMUNICACIONES.

SOFTWARE: CONSISTE DE LOS PROGRAMAS DE INSTRUCCIONES EN LENGUAJE DE MAQUINA Y DATOS QUE SON INTERPRETADOS POR EL HARDWARE. ALGUNOS TIPOS COMUNES DE SOFTWARE SON COMPILADORES, ENSAMBLADORES, CARGADORES, EDITORES ENCADENADORES (LINKAGE EDITORS), CARGADORES ENCADENANTES (LINKING LOADERS), PROGRAMAS DE APLICACION DEL USUARIO, SISTEMAS DE ADMINISTRACION DE BASES DE DATOS (DBMS), SISTEMAS DE COMUNICACION DE DATOS Y SISTEMAS OPERATIVOS.

FIRMWARE: CONSISTE DE PROGRAMAS EN MICROCODIGO EJECUTADOS DESDE ALMACENAMIENTO DE CONTROL DE MUY ALTA VELOCIDAD. PROGRAMAS OBJETO COMUNMENTE UTILIZADOS PUESTOS EN (ROM S Y PROM S) ES TAMBIEN LLAMADO FIRMWARE.

2.2 ALMACENAMIENTO INTERLEAVING

ES USADO PARA AUMENTAR LA VELOCIDAD DE ACCESO A ALMACENAMIENTO PRIMARIO. ORDINARIAMENTE MIENTRAS ALGUNA DE LAS LOCALIZACIONES EN UN BANCO DE ALMACENAMIENTO PRIMARIO ESTA SIENDO ACCESADO, NINGUNA OTRA REFERENCIA SE PUEDE ESTAR HACIENDO. ALMACENAMIENTO INTERLEAVING COLOCA LOCALIZACIONES DE ALMACENAMIENTO ADYACENTES EN DIFERENTES BANCOS DE ALMACENAMIENTO DE TAL MANERA QUE SE PUEDAN HACER MULTIPLES REFERENCIAS AL MISMO TIEMPO.

2.3 REGISTRO DE RELOCALIZACION

UN REGISTRO DE RELOCALIZACION PERMITE QUE LOS PROGRAMAS SEAN RELOCALIZADOS DINAMICAMENTE. LA DIRECCION BASE DEL PROGRAMA EN ALMACENAMIENTO PRINCIPAL ES COLOCADA EN EL REGISTRO DE RELOCALIZACION Y SU CONTENIDO ES SUMADO A CADA DIRECCION GENERADA POR UN PROGRAMA CORRIENDO. EL USUARIO PUEDE PROGRAMAR COMO SI EL PROGRAMA EMPEZARA EN LA DIRECCION CERO. AL TIEMPO DE EJECUCION TODAS LAS REFERENCIAS A DIRECCIONES INVOLUCRAN EL REGISTRO DE RELOCALIZACION; ESTO PERMITE AL PROGRAMA OCUPAR LOCALIZACIONES DIFERENTES A LAS QUE SE LES HABIAN ASIGNADO EN LA TRADUCCION.

2.4 INTERRUPCIONES Y POLEO (POLLING)

TECNICA PARA PERMITIR A UNA UNIDAD CHECAR EL ESTADO (STATUS) DE OTRA UNIDAD FUNCIONANDO INDEPENDIENTEMENTE; LA PRIMERA UNIDAD CHECA SI LA SEGUNDA UNIDAD ESTA EN CIERTO STATUS Y SI NO LO ESTA ENTONCES LA PRIMERA UNIDAD PROCEDE CON LO QUE ESTABA HACIENDO. POLLING PUEDE SER UNA OPERACION DE ALTO OVERHEAD.

LAS INTERRUPCIONES PERMITEN A UNA UNIDAD GANAR LA ATENCION INMEDIATA DE OTRA DE TAL MANERA QUE LA PRIMERA PUEDA REPORTAR UN CAMBIO DE STATUS. LA INTERRUPCION CAUSA QUE EL STATUS DE LA UNIDAD INTERRUMPIDA SEA (CAMBIADO) SALVADO ANTES DE QUE LA INTERRUPCION SEA PROCESADA. DESPUES LA INTERRUPCION ES PROCESADA Y EL STATUS DE LA UNIDAD INTERRUMPIDA ES RESTAURADO.

2.5 BUFFERIZACION.

UN BUFFER ES UNA AREA DE ALMACENAMIENTO PRIMARIO PARA ALMACENAR DATOS DURANTE TRANSFERENCIAS DE I/O. AL ESTAR EFECTUANDOSE UNA TRANSFERENCIA DE I/O, SU VELOCIDAD DEPENDE DE MUCHOS FACTORES RELACIONADOS AL HARDWARE DE I.O, PERO NORMALMENTE NO RELACIONADOS A LA OPERACION DEL PROCESADOR. EN LA ENTRADA POR EJEMPLO, EL DATO ES COLOCADO EN EL BUFFER POR UN CANAL DE I/O; CUANDO LA TRANSFERENCIA TERMINA EL DATO PUEDE SER ACCESADO POR EL PROCESADOR.

CON ENTRADA DE BUFFER SENCILLO (SINGLE-BUFFERED), EL CANAL DEPOSITA EL DATO EN EL BUFFER, EL PROCESADOR PROCESA EL DATO, EL CANAL DEPOSITA EL SIGUIENTE DATO, ETC. MIENTRAS EL CANAL ESTA DEPOSITANDO DATOS NINGUN PROCESO DE DATOS PUEDE OCURRIR MIENTRAS EL DATO ESTA SIENDO PROCESADO, DATOS ADICIONALES NO PUEDEN SER DEPOSITADOS.

UN SISTEMA DE DOBLE BUFFERIZACION PERMITE MEZCLAR OPERACIONES DE I/O CON PROCESAMIENTO; MIENTRAS EL CANAL ESTA DEPOSITANDO DATOS EN UN BUFFER, EL PROCESADOR PUEDE ESTAR PROCESANDO DATOS EN EL OTRO BUFFER. CUANDO EL PROCESADOR TERMINA DE PROCESAR DATOS EN UN BUFFER, PROCESA LOS DATOS DEL SEGUNDO BUFFER MIENTRAS EL CANAL DEPOSITA NUEVOS DATOS EN EL PRIMERO. ESTE USO ALTERNO DE LOS BUFFER ES LLAMADO ALGUNAS VECES BUFFERIZACION FLIP-FLOP.

2.6 DISPOSITIVOS PERIFERICOS.

LOS DISPOSITIVOS PERIFERICOS PERMITEN EL ALMACENAMIENTO DE CANTIDADES MASIVAS DE INFORMACION AFUERA DEL ALMACENAMIENTO PRINCIPAL DE LA COMPUTADORA. LOS MANEJADORES DE CINTA (TAPE DRIVES) SON DISPOSITIVOS INHERTEMENTE SECUENCIALES QUE LEEN Y ESCRIBEN DATOS EN UNA LARGA TIRA DE CINTA MAGNETICA. LAS CINTAS PUEDEN MEDIR HASTA 3600 PIES EN UN CARRETE DE 12 PULGADAS. LA INFORMACION PUEDE SER GRABADA EN CINTA A VARIAS DENSIDADES. LOS PRIMEROS SISTEMAS GRABABAN 200 CARACTERES

POR PULGADA (CPI) DE CINTA. DESPUES SE HICIERON POPULARES LAS DENSIDADES DE 556 CPI, 800, 1600 Y AHORA LAS DE 6250. DENSIDADES MAYORES SERAN SOPORTADAS EN SISTEMAS FUTUROS.

EL DISPOSITIVO PERIFERICO MAS SIGNIFICANTE PARA EL S.O. ES EL MANEJADOR DE DISCO MAGNETICO (MAGNETIC DISK DRIVE).

LOS DISCOS SON DISPOSITIVOS DE ACCESO DIRECTO QUE PERMITEN REFERENCIAR CAMPOS DE DATOS INDIVIDUALES SIN LA NECESIDAD DE BUSCAR TODOS LOS CAMPOS EN EL DISCO LOS PRIMEROS DISPOSITIVOS EN DISCO ALMACENABAN VARIOS MILLONES DE CARACTERES, LAS UNIDADES DE HOY ALMACENAN HASTA UN BILLON DE CARACTERES. LAS UNIDADES A SER DISENADAS EN EL FUTURO PROPORCIONARAN CAPACIDADES AUN MAYORES.

2.7 PROTECCION DE ALMACENAMIENTO (STORAGE PROTECTION).

ES ESENCIAL EN SISTEMAS COMPUTACIONALES DE MULTIPLES USUARIOS. YA QUE LIMITA EL RANGO DE DIRECCIONES QUE UN PROGRAMA PUEDE REFERENCIAR. STORAGE PROTECTION PUEDE SER IMPLEMENTADA PARA UN PROGRAMA EN UN BLOCK CONTIGUO DE LOCALIZACIONES DE ALMACENAMIENTO (STORAGE) POR REGISTROS FRONTERA (BOUNDS REGISTERS) QUE DEFINEN LA DIRECCION INFERIOR Y SUPERIOR DEL BLOCK DE ALMACENAMIENTO. CUANDO UN PROGRAMA SE EJECUTA, TODAS LAS DIRECCIONES REFERENCIADAS SON CHECADAS PARA VER SI ESTAN ENTRE LAS DIRECCIONES EN LOS REGISTROS. TAMBIEN PUEDE SER IMPLEMENTADA CON EL USO DE LLAVES DE PROTECCION LIGADAS A AREAS EN ALMACENAMIENTO PRIMARIO; UN PROGRAMA PUEDE REFERENCIAR DIRECCIONES SOLO EN AQUELLAS AREAS CON LLAVES QUE SEAN IGUALES A LAS LLAVES DEL PROGRAMA.

2.8 MARCADORES DE TIEMPO (TIMERS) Y RELOJES (CLOCKS).

UN MARCADOR DE INTERVALOS DE TIEMPO ES UTIL EN SISTEMAS DE MULTIUSUARIOS PARA PREVENIR A UN USUARIO DE MONOPOLIZAR UN PROCESADOR. DESPUES DE UN CIERTO INTERVALO, EL TIMER GENERA UNA INTERRUPCION PARA GANAR LA ATENCION DEL PROCESADOR Y SER ASIGNADO A OTRO USUARIO.

UN RELOJ DE TIEMPO DEL DIA PROPORCIONA UN MEDIO PARA LA COMPUTADORA DE LLEVAR EL CONTROL DE TIEMPO EN INCREMENTOS TAN FINOS O MAS FINOS QUE MILLONESIMAS DE SEGUNDO.

2.9 OPERACIONES EN LINEA (ON-LINE) Y FUERA DE LINEA (OFF- LINE) PROCESADORES SATELITE.

ALGUNOS PERIFERICOS HAN SIDO EQUIPADOS PARA OPERACION ON-LINE EN LA CUAL SON CONECTADAS AL PROCESADOR U OPERACION OFF-LINE EN LA CUAL SON CORRIDAS POR UNIDADES DE CONTROL NO CONECTADAS AL SISTEMA COMPUTACIONAL CENTRAL.

LAS UNIDADES DE CONTROL OFF-LINE HACEN POSIBLE MANEJAR

DISPOSITIVOS PERIFERICOS SIN COLOCAR CARGA DIRECTAMENTE EN EL PROCESADOR. CINTA A TARJETA, TARJETA-CINTA Y OPERACIONES CINTA-IMPRESION SON FRECUENTMENTE HECHAS POR UNIDADES OFF-LINE.

2.10 CANALES DE ENTRADA/SALIDA (INPUT/OUTPUT).

CUANDO LAS DEMANDAS SOBRE LOS PRIMEROS SISTEMAS SE INCREMENTARON PARTICULARMENTE PROCESAMIENTO DE DATOS COMERCIALES, LOS SISTEMAS TENDIERON A UNA CUOTA EN I/O. MIENTRAS I/O ESTABA ACTIVO, LOS PROCESADORES ESTABAN OBSTRUIDOS MANEJANDO I/O. EN ALGUNOS SISTEMAS SOLO UNA OPERACION DE I/O PODIA SER MANEJADA A LA VEZ.UN MEJORAMIENTO IMPORTANTE PARA MODIFICAR ESTA SITUACIOB FUE EL DESARROLLO DE CANALES.

UN CANAL ES UN SISTEMA COMPUTACIONAL DE PROPOSITO ESPECIAL DEDICADO A MANEJAR I/O INDEPENDIENTEMENTE DEL PROCESADOR PRINCIPAL DEL SISTEMA.

UN CANAL PUEDE ACCESAR MEMORIA PRIMARIA DIRECTAMENTE PARA ALMACENAR O RECUPERAR INFORMACION.

LOS PRIMEROS SISTEMAS MANEJARON COMUNICACION ENTRE PROCESADORES Y CANALES POR INSTRUCCIONES TALES COMO:

- BIFURCAR SI UN CANAL ESTA EN OPERACION
- ESPERAR A QUE SE TERMINE UN COMANDO
- ALMACENAR EL CONTENIDO DE LOS REGISTROS DE CONTROL DE CANAL EN E DEL PROCESADOR

PRINCIPAL DEL SISTEMA.

UN CANAL PUEDE ACCESAR MEMORIA PRIMARIA DIRECTAMENTE PARA ALMACENAR O RECUPERAR INFORMACION.

LOS PRIMEROS SISTEMAS MANEJARON COMUNICACION ENTRE PROCESADORES Y CANALES POR INSTRUCCIONES TALES COMO:

- BIFURCAR SI UN CANAL ESTA EN OPERACION
- ESPERAR A QUE SE TERMINE UN COMANDO
- ALMACENAR EL CONTENIDO DE LOS REGISTROS DE CONTROL DE CANAL EN MEMORIA PRINCIPAL PARA CONSULTAS SUBSECUENTES DEL PROCESADOR.

EN LOS SISTEMAS MANEJADOS POR INTERRUPTS, UN PROCESADOR EJECUTA UNA INSTRUCCION START I.O PARA INICIAR UNA TRANSFERENCIA I/O SOBRE UN CANAL; EL CANAL GENERA UNA INTERRUPCION DE TERMINACION DE I/O PARA INFORMAR AL PROCESADOR.

EL SIGNIFICADO REAL DE LOS CANALES ES QUE INCREMENTAN LA ACTIVIDAD DE A LA VEZ; UN CANAL MULTIPLEXOR DE BYTES INTERCALA LAS TRANSMICIONES DE DISPOSITIVOS LENTOS COMO TERMINALES, LECTORAS, PERFORADORAS, IMPRESORAS Y LINEAS DE COMUNICACION DE ALTA VELOCIDAD. UN CANAL MULTIPLEXOR DE BLOQUES INTERCALA LA TRANSMICION DE VARIOS DISPOSITIVOS DE ALTA VELOCIDAD TALES COMO IMPRESORAS LASERS Y MANEJADORES DE DISCOS (DISK DRIVES).

2.11 CICLO STEALING.

UN PUNTO DE CONFLICTO ENTRE CANALES Y EL PROCESADOR ES EL ACCESO A MEMORIA PRIMARIA. PUESTO QUE UN SOLO ACCESO (A UN BANCO DE MEMORIA PRINCIPAL) PUEDE EFECTUARSE A LA VEZ, Y DADO QUE ES POSIBLE QUE LOS CANALES Y EL PROCESADOR PUEDEN DESEAR EL ACCESO A MEMORIA PRINCIPAL SIMULTANEAMENTE, A LOS CANALES SE LES DA PRIORIDAD. ESTO ES LLAMADO CICLO STEALING. EL CANAL LITERALMENTE ROBA O PLAGIA CICLOS DE ALMACENAMIENTO AL PROCESADOR. LOS CANALES SOLO USAN UN PEQUENO PORCENTAJE DE LOS CICLOS, PERO DANDOLES PRIORIDAD DE ESTA MANERA CAUSA UNA MEJOR UTILIZACION DE LOS DISPOSITIVOS DE I/O. ESTE TIPO DE LOGICA HA SIDO TRANSPORTADA A LOS S.O.
PROGRAMAS DE COTA DE I/O (I/O BOUNDS) SON GENERALMENTE DADOS PRIORIDAD SOBRE PROGRAMAS DE COTA DE PROCESADOR (PROCESSOR-BOUND) POR LOS MECANISMOS DE ASIGNACION DEL SITEMA OPERATIVO.

EXPLICAME

? ## 2.12 DIRECCIONAMIENTO BASE-MAS DESPLAZAMIENTO.

CUANDO HUBO NECESIDAD DE UNA CAPACIDAD DE ALMACENAMIENTO MAYOR, LAS ARQUITECTURAS FUERON MODIFICADAS PARA ACOMODAR UN RANGO MUY GRANDE DE DIRECCIONES. UN SISTEMA DISENADO PARA SOPORTAR 16 MB (M=1048576) REQUERIA DIRECCIONES DE 24 BITS.
INCORPORAR DIRECCIONES DE TAL LONGITUD EN LAS INSTRUCCIONES DE MAQUINA CON INSTRUCCIONES DE UNA SOLA DIRECCION SERIA COSTOSO; EN UNA MAQUINA CON INSTRUCCIONES DE MULTIPLES DIRECCIONES SERIA INTOLERABLE.
ASI ES QUE PARA LOGRAR UN GRAN RANGO DE DIRECCIONES, SE USA EL DIRECCIONAMIENTO BASE-MAS-DESPLAZAMIENTO EN EL CUAL TODAS LAS DIRECCIONES SON SUMADAS AL CONTENIDO DE UN REGISTRO BASE. ESTE ESQUEMA TIENE LA VENTAJA ADICIONAL DE HACER PROGRAMAS INDEPENDIENTES DE LA UBI- CACION, UNA PROPIEDAD PARTICULARMENTE VALIOSA PARA PROGRAMAS EN AMBIENTES DE MULTIUSUARIOS EN EL CUAL UN PROGRAMA PUEDE TENER QUE SER COLOCADO EN DIFERENTES UBICACIONES EN MEMORIA PRINCIPAL CADA VEZ QUE ES CARGADO.

2.13 ESTADO PROBLEMA, ESTADO SUPERVISOR, INSTRUCCIONES PRIVILEGIADAS.

LOS SISTEMAS COMPUTACIONALES GENERALMENTE TIENEN VARIOS ESTADOS DIFERENTES DE EJECUCION. NORMALMENTE CUANDO LA MAQUINA ESTA EN UN ESTADO PARTICULAR SOLO UN SUBCONJUNTO DE SUS INSTRUCCIONES ES EJECUTABLE POR EL PROGRAMA EN EJECUCION. PARA PROGRAMAS DE USUARIO, EL SUBCONJUNTO DE INSTRUCCIONES QUE EL USUARIO PUEDE EJECUTAR EN EL ESTADO PROBLEMA INCLUYE, POR EJEMPLO, LA EJECUCION DIRECTA DE INSTRUCCIONES DE ENTRADA/SALIDA (I/O) A UN PROGRAMA DEL

USUARIO QUE SE LE PERMITIERA EFECTUAR LO ARBITRARIO PODRIA LISTAR LOS PASSWORDS, IMPRIMIR LA INFORMACION DE CUALQUIER OTRO USUARIO, O DESTRUIR EL SISTEMA OPERATIVO. EL SISTEMA OPERATIVO GENERALMENTE CORRE CON EL STATUS DE USUARIO DE MAYOR CONFIANZA (MOST TRUSTED USER) EN UN ESTADO SUPERVISOR; TIENE ACCESO A TODAS LAS INSTRUCCIONES EN EL CONJUNTO DE INSTRUCCIONES DE MAQUINA. TAL DISCUSION DE ESTOS PROBLEMAS ESTADO SUPERVISOR HA SIDO ADECUADO PARA UNA DE LOS SISTEMAS MODERNOS DE COMPUTACION.

EN EL CASO DE SISTEMAS ALTAMENTE SEGUROS, SIN EMBARGO, ES DESEABLE TENER MAS DE 2 ESTADOS. ESTO DA UNA GRANULARIDAD DE PROTECCION MAS FINA. TAMBIEN PERMITE ACCESO GARANTIZADO POR EL PRINCIPIO DE MINIMO PRIVILEGIO EN EL CUAL CUALQUIER USUARIO PARTICULAR DEBE TENER GARANTIZADA LA CANTIDAD MINIMA DE PRIVILEGIOS Y ACCESOS QUE NECESITA PARA DESARROLLAR SUS TAREAS.

ES INTERESANTE QUE A MEDIDA QUE LAS ARQUITECTURAS COMPUTACIONALES HAN EVOLUCIONADO EL NUMERO DE INSTRUCCIONES PRIVILEGIADAS, ES DECIR AQUELLAS INSTRUCCIONES NO ACCESIBLES EN ESTADO PROBLEMA, HAN TENDIDO A INCREMENTARSE. ESTO INDICA UNA TENDENCIA DEFINIDA HACIA LA INCORPORACION DE MAS FUNCIONES DE LOS SISTEMAS OPERATIVOS EN HARDWARE.

ALGUNOS MICROPROCESADORES YA TIENEN SISTEMAS OPERATIVOS COMPLETOS EN FIRMWARE; MUCHOS TIENEN MUCHA FUNCIONALIDAD DEL SISTEMA OPERATIVO EN HARDWARE.

2.14 ALMACENAMIENTO VIRTUAL.

LOS SISTEMAS DE ALMACENAMIENTO VIRTUAL PERMITEN A LOS PROGRAMAS REFERENCIAR DIRECCIONES QUE NO NECESITAN CORRESPONDER A DIRECCIONES REALES DISPONIBLES EN ALMACENAMIENTO PRIMARIO. LAS DIRECCIONES VIRTUALES DESARROLLADAS POR LOS PROGRAMAS EN EJECUCION SON TRADUCIDAS DINAMICAMENTE (A TIEMPO DE EJECUCION) POR EL HARDWARE EN DIRECCIONES DE INSTRUCCIONES Y DATOS EN MEMORIA PRINCIPAL. LOS SISTEMAS DE ALMACENAMIENTO VIRTUAL PERMITEN A LOS PROGRAMAS REFERENCIAR ESPACIOS DE DIRECCIONES MAYORES QUE LOS DISPONIBLES EN MEMORIA PRINCIPAL. PERMITEN A LOS USUARIOS CREAR PROGRAMAS INDEPENDIENTES (EN SU MAYOR PARTE) DE LAS RESTRICCIONES DE MM, Y FACILITA LA OPERACION DE SISTEMAS COMPARTIDOS DE MULTIUSUARIOS.

ESTOS SISTEMAS USAN LAS TECNICAS DE PAGINACION DE BLOCKS DE TAMAÑO FIJO DE DATOS ENTRE MEMORIA PRINCIPAL Y ALMACENAMIENTO SECUNDARIO, Y SEGMENTACION EL CUAL IDENTIFICA UNIDADES LOGICAS DE PROGRAMAS Y DATOS PARA FACILITAR EL CONTROL DE ACCESO Y COMPARTICION. ESTAS TECNICAS SON ALGUNAS VECES USADAS INDIVIDUALMENTE Y OTRAS EN FORMA COMBINADA.

2.15 MULTIPROCESAMIENTO.

EN SISTEMAS DE MULTIPROCESAMIENTO VARIOS PROCESOS COMPARTEN UNA AREA DE ALMACENAMIENTO PRIMARIO EN FORMA COMUN Y UN SOLO SISTEMA OPERATIVO. MULTIPROCESAMIENTO INTRODUCE EL POTENCIAL PARA CIERTOS TIPOS DE CONFLICTOS QUE NO OCURREN EN LOS SISTEMAS DE UN PROCESADOR.
ES NECESARIO SECUENCIALIZAR EL ACCESO A UNA LOCALIZACION COMPARTIDA PARA QUE 2 PROCESADORES NO INTENTEN MODIFICARLA AL MISMO TIEMPO, POSIBLEMENTE DANANDO SU CONTENIDO. LA SECUENCIALIZACION TAMBIEN ES NECESARIA EN SISTEMAS UNIPROCESADORES.

2.16 MEMORIA DE ACCESO DIRECTO (DMA)

UNA CLAVE PARA OBTENER BUENA EJECUCION (PERFORMANCE) EN LOS SISTEMAS COMPUTACIONALES ES MINIMIZAR EL NUMERO DE INTERRUPTS QUE OCURREN MIENTRAS SE EJECUTA UN PROGRAMA. DMA REQUIERE SOLO UN INTERRUPT PARA CADA BLOCK DE CARACTERES TRANSFERIDOS EN UNA OPERACION DE I/O, POR LO QUE ES SIGNIFICANTEMENTE MAS RAPIDO QUE EL METODO EN EL CUAL EL PROCESADOR ES INTERRUMPIDO POR CADA CARACTER TRANSFERIDO. UNA VEZ QUE UNA OPERACION DE I/O ES INICIALIZADA LOS CARACTERES SON TRANSFERIDOS A MEMORIA PRINCIPAL EN BASE AL CICLO STEALING (EL CANAL TEMPORALMENTE USURPA LA RUTA DEL PROCESADOR PARA ALMACENAR MIENTRAS UN CARACTER ESTA SIENDO TRANSFERIDO, DESPUES EL PROCESADOR CONTINUA SIN OPERACION.
CUANDO UN DISPOSITIVO ESTA LISTO PARA TRANSMITIR UN CARACTER DEL BLOCK, INTERRUMPE AL PROCESADOR. PERO CON DMA EL ESTADO DEL PROCESADOR NO TIENE QUE SER SALVADO, EL PROCESADOR ES RETARDADO MAS QUE INTERRUMPIDO. BAJO EL CONTROL DE HARDWARE ESPECIAL EL CARACTER ES TRANSFERIDO. CUANDO LA TRANSFERENCIA TERMINA, EL PROCESADOR RESUME ESA OPERACION.

2.17 PARALELISMO (PIPELINING).

—¿Como trabaja la arquitectura para lograr paralelismo?

ES UNA TECNICA DE HARDWARE USADA EN SISTEMAS COMPUTACIONALES DE ALTO DESARROLLO PARA EXPLOTAR CIERTOS TIPOS DE PARALELISMO EN EL PROCESAMIENTO DE INSTRUCCIONES. SIMPLEMENTE, EL PROCESADOR ES ARREGLADO COMO UNA LINEA DE PRODUCCION EN UNA FABRICA; VARIAS INSTRUCCIONES PUEDEN ESTAR EN DIFERENTES ETAPAS DE EJECUCION SIMULTANEAMENTE. ESTE EMPALME (OVERLAP) REQUIERE HARDWARE MAS EXTENSIVO PERO REDUCE EL TIEMPO TOTAL DE EJECUCION DE UNA SECUENCIA DE INSTRUCCIONES.

2.18 JERARQUIA DE ALMACENAMIENTO.

LOS SISTEMAS DE HOY TIENEN VARIOS NIVELES: MEMORIA PRINCIPAL, MEMORIA SECUNDARIA Y CACHE. LAS

INSTRUCCIONES Y DATOS DEBEN ESTAR EN MEMORIA PRINCIPAL (MP) PARA SER REFERENCIADO POR UN PROGRAMA CORRIENDO.

MEMORIA SECUNDARIA (MS): CINTAS, DISCOS, TARJETAS Y OTRAS MEDIOS.

CACHE: MEMORIA RAPIDA DISENA PARA INCREMENTAR LA VELOCIDAD DE EJECUCION DE LOS PROGRAMAS ES TRANSPARENTE AL USUARIO. LA PORCION ACTUAL DE UN PROGRAMA ES COLOCADA EN CACHE.

LOS NIVELES CREAN UNA JERARQUIA DE ALMACENAMIENTO: LOS NIVELES DE CACHE A MEMORIA SECUNDARIA DECREMENTAN COSTO Y VELOCIDAD Y AUMENTAN CAPACIDAD.

LA MEMORIA ES DIVIDIDA EN BYTES (CARACTERES) O PALABRAS (CONSISTE EN UN NUMERO FIJO DE BYTES). CADA LOCALIZACION TIENE UNA DIRECCION Y EL CONJUNTO DE TODAS LAS DIRECCIONES DE UN PROGRAMA ES LLAMADO ESPACIO DE DIRECCIONES.

2.19 SOFTWARE.

CONSISTE DE LOS PROGRAMAS DE INSTRUCCIONES Y DATOS QUE DEFINEN AL HARDWARE Y LOS ALGORITMOS PARA RESOLVER LOS PROBLEMAS. HAY UNA AMPLIA VARIEDAD DE LENGUAJES DE PROGRAMACION.

2.20 PROGRAMACION EN EL LENGUAJE MAQUINAL.

EL LENGUAJE MAQUINAL ES EL LENGUAJE DE PROGRAMACION QUE UNA COMPUTADORA PUEDE ENTENDER DIRECTAMENTE. CADA INSTRUCCION EN LENGUAJE DE MAQUINA ES INTERPRETADA POR EL HARDWARE QUE EFECTUA LAS FUNCIONES INDICADAS. LAS INSTRUCCIONES EN LENGUAJE DE MAQUINA HAN SIDO GENERALMENTE PRIMITIVAS, EL ARREGLO DE ESAS INSTRUCCIONES EN PROGRAMAS EN LENGUAJE DE MAQUINA ES LO QUE PERMITE LA ESPECIFICACION DE ALGORITMOS UTILES. EL REPERTORIO DE INSTRUCCIONES EN LENGUAJE DE MAQUINA, ACTUALMENTE INCLUYE APTITUDES MUY PODEROSAS.

EL LENGUAJE DE MAQUINA SE DICE QUE ES DEPENDIENTE DE LA MAQUINA; UN PROGRAMA EN LENGUAJE DE MAQUINA ESCRITO PARA UNA COMPUTADORA NO PUEDE CORRER ORDINARIAMENTE EN OTRA A MENOS QUE SU LENGUAJE DE MAQUINA SEA IDENTICO (O MAYOR) QUE EL DE LA OTRA. OTRA INDICACION DE LA DEPENDENCIA DE MAQUINA ES LA BONDAD DE LAS INSTRUCCIONES MISMAS; INSTRUCCIONES EN LENGUAJE DE MAQUINA NOMBRAN REGISTROS ESPECIFICOS DEL SISTEMA COMPUTACIONAL Y PROCESAN DATOS EN LA FORMA FISICA EN LA CUAL LOS DATOS EXISTEN EN EL SISTEMA COMPUTACIONAL. LA MAYORIA DE LOS PRIMEROS SISTEMAS FUERON PROGRAMADOS DIRECTAMENTE EN LENGUAJE DE MAQUINA; HOY SON POCOS LOS PROGRAMAS EN LENGUAJE DE MAQUINA.

2.21 ENSAMBLADORES Y MACROPROCESADORES.

LA PROGRAMACION EN LENGUAJE DE MAQUINA CONSUME TIEMPO

ESTA PROPENSA A ERRORES.
LOS LENGUJES DE ENSAMBLADO FUERON DESARROLLADOS PARA ACELERAR EL PROCESO DE PROGRAMACION Y REDUCIR ERRORES DE CODIFICACION. USAN ABREVIATURAS SIGNIFICATIVAS Y PALABRAS PARA REEMPLAZAR LOS STRINGS DE NUMEROS USADOS PARA ESCRIBIR PROGRAMAS EN LENGUAJE DE MAQUINA. PERO LOS PROGRAMAS EN LENGUAJE DE ENSAMBLADO NO SON ENTENDIBLES DIRECTAMENTE POR LA COMPUTADORA LOS PROGRAMAS DEBEN SER TRADUCIDOS AL LENGUAJE DE MAQUINA ESTA TRADUCCION ES EFECTUADA POR UN PROGRAMA LLAMADO ENSAMBLADOR.

LOS LENGUAJES DE ENSAMBLADO SON TAMBIEN DEPENDIENTES DE LA MAQUINA. SUS INSTRUCCIONES CORRESPONDEN 3 A 1 CON LAS INSTRUCCIONES EN UN PROGRAMA EN LENGUAJE DE MAQUINA. PARA ACELERAR EL PROCESO DE CODIFICACION DE PROGRAMAS EN ENSAMBLADO MACROPROCESADORES (MACROS) FUERON DESARROLLADOS E INCORPORADOS EN LOS ENSAMBLADORES.

UNA MACROINSTRUCCION ES ESCRITA POR EL PROGRAMADOR PARA INDICAR LA EJECUCION DE UNA TAREA QUE REQUIERE VARIAS INSTRUCCIONES EN LENGUAJE DE ENSAMBLADO. CUANDO EL PROCESADOR DE MACROINSTRUCCIONES LEE UNA MACRO DURANTE LA TRADUCCION, HACE UNA EXPANSION DE LA MACRO, GENERA UNA SERIE DE INSTRUCCIONES EN LENGUAJE DE ENSAMBLADO CORRESPONDIENTE A LA MACROINSTRUCCION. EL PROCESO DE PROGRAMACION ES ACELERADO PORQUE EL PROGRAMADOR ESCRIBE MENOS INSTRUCCIONES PARA DEFINIR EL MISMO ALGORITMO.

2.22 COMPILADORES.

LA TENDENCIA A HACER INSTRUCCIONES MAS PODEROSAS CONDUJO AL DESARROLLO DE ALGUNOS MACROPROCESADORES Y MACROLENGUAJES MAS SOFISTICADOS PARA APOYAR A LOS PROGRAMADORES DE LENGUAJE DE ENSAMBLADO. PERO AUMENTAR LOS ENSAMBLADORES AGREGANDO MACROPROCESADORES AUN NO RESUELVE EL PROBLEMA DE DEPENDENCIA DE MAQUINA, LO CUAL CONDUCE AL DESARROLLO DE LOS LENGUAJES DE ALTO NIVEL, QUE PERMITEN AL USUARIO ESCRIBIR PROGRAMAS EN UNA MANERA INDEPENDIENTE DE LA MAQUINA.

LOS USUARIOS SE INTERESAN EN LA COMPUTADORA COMO UNA HERRAMIENTA; LOS LENGUAJES DE ALTO NIVEL PERMITEN AL USUARIO DEDICARSE AL PROBLEMA UNICO DE SUS APLICACIONES SIN INTERESARLE DEPENDENCIA ALGUNA DEL HARDWARE, LO QUE FACILITA EL PROCESO DE PROGRAMACION, HACE LOS PROGRAMAS TRANSPORTABLES Y PERMITE HACER SISTEMAS DE APLICACION SIN CONOCER LA ESTRUCTURA INTERNA DEL SISTEMA.

LOS LENGUAJES DE ALTO NIVEL SON TRADUCIDOS POR PROGRAMAS LLAMADOS COMPILADORES. COMPILADORES Y ENSAMBLADORES SON REFERENCIADOS COMO TRADUCTORES. EN EL PROCESO DE TRADUCCION, EL PROGRAMA ESCRITO POR EL USUARIO QUE ES LA ENTRADA AL TRADUCTOR ES LLAMADO PROGRAMA FUENTE; EL PROGRAMA EN LENGUAJE DE MAQUINA PRODUCIDO POR EL TRADUCTOR ES EL PROGRAMA OBJETO O PROGRAMA BLANCO.

¿EXISTE DIFERENCIA ENTRE COMPILADORES Y ENSAMBLADORES?
- ¿CUÁLES?

2.23 SISTEMA DE CONTROL INPUT/OUTPUT (IOCS).

LOS PROGRAMAS DETALLADOS DE CANAL, NECESARIOS PARA CONTROLAR I/O Y LAS RUTINAS PARA COORDINAR LA OPERACION DE CANALES Y PROCESADORES SON COMPLEJOS. EL DESARROLLO DE UN PROGRAMA SUPERVISOR PARA MANEJAR LAS COMPLEJIDADES DE I/O REMOVIO ESTA CARGA DEL PROGRAMADOR DE APLICACIONES. ESTE PROGRAMA SUPERVISOR ES LLAMADO IOCS.

EN LOS 50'S LOS USARIOS PODIAN INCLUIR EL CODIGO FUENTE DE IOCS CON SUS ESTATUTOS EN LENGUAJE DE ENSAMBLADO. EL PAQUETE IOCS YA ESCRITO Y DEPURADO, ERA REENSAMBLADO COMO PARTE DE CADA PROGRAMA INDIVIDUAL; SIN EMBARGO ESTO ALARGABA EL TIEMPO DE TRADUCCION DEL PROGRAMA, POR LO TANTO EN MUCHOS SISTEMAS LAS RUTINAS IOCS PREENSAMBLADAS FUERON USADAS CON FRECUENCIA. EL PROGRAMADOR EN LENGUAJE DE ENSAMBLADO QUE REFERENCIABA LAS LOCALIZACIONES DE RUTINAS CLAVE EN EL CODIGO IOCS PREENSAMBLADO.

OTRO PROBLEMA CON EL CONCEPTO IOCS FUE EL HECHO QUE EL PAQUETE COMPLETO IOCS OCUPABA UNA PORCION SIGNIFICANTE DE MEMORIA PRINCIPAL DEJANDO POCO ESPACIO PARA EL CODIGO DE APLICACIONES. ALGUNOS USUARIOS SOBRE ESCRIBIAN EN CIERTAS PORCIONES DE IOCS QUE NO ERAN NECESARIAS. OTROS ESCRIBIAN SUS PROPIOS PAQUETES.

FINALMENTE, LOS USUARIOS VIERON LA IMPORTANCIA DE DEJAR EL CONTROL DE I/O A IOCS Y FUERON SIMPLEMENTE FORZADOS A AGREGAR MEMORIA PRIMARIA MAS CARA A SUS SISTEMAS COMPUTACIONALES. ESTA TENDENCIA HA SIDO FIRMEMENTE ESTABLECIDA Y LOS S.O. HAN IDO MAS Y MAS HACIA CODIGO ORIENTADO AL SISTEMA DE TAL MANERA QUE LOS DESARROLLADORES DE APLICACIONES PUEDAN CONCENTRARSE EN PRODUCIR CODIGO ORIENTADO A APLICACIONES. ESTO HA CAUSADO QUE LOS S.O. REQUIERAN MAS MEMORIA PRIMARIA PERO AFORTUNADAMENTE EL COSTO DE ALMACENAMIENTO PRIMARIO HA IDO DISMINUYENDO.

2.24 SPOOLING.

EN SPOOLING (SIMULTANEOUS PERIPHERAL OPERATION ON LINE) UN DISPOSITIVO DE ALTA VELOCIDAD, TAL COMO UN DISCO ES INTERPUESTO ENTRE UN PROGRAMA CORRIENDO Y UN DISPOSITIVO DE BAJA VELOCIDAD, INVOLUCRADAS CON EL PROGRAMA EN I.O. EN LUGAR DE ESCRIBIR LINEAS DIRECTAMENTE A LA IMPRESORA DE LINEAS, ESTAS SON ESCRITAS AL DISCO. EL PROGRAMA PUEDE CORRER PARA TERMINAR MAS RAPIDO Y OTROS PROGRAMAS PUEDAN SER INICIADOS MAS PRONTO. CUANDO LA IMPRESORA ESTA DISPONIBLE, LAS LINEAS PUEDEN SER IMPRESAS. EL NOMBRE SPOOLING ES MUY APROPIADO PARA ESTE PROCEDIMIENTO.

2.25 LENGUAJES ORIENTADOS A PROCEDIMIENTOS VS LENGUAJES ORIENTADOS A PROBLEMAS.

LOS LENGUAJES DE ALTO NIVEL SON ORIENTADOS A PROCEDIMIENTOS U ORIENTADOS A PROBLEMAS. LOS LENGUAJES DE ALTO NIVEL ORIENTADOS A PROCEDIMIENTOS SON LENGUAJES DE PROPOSITO GENERAL QUE PUEDEN SER USADOS PARA RESOLVER UNA GRAN VARIEDAD DE PROBLEMAS. LOS LENGUAJES ORIENTADOS A PROBLEMAS SON ESPECIFICAMENTE PARA RESOLVER TIPOS PARTICULARES DE PROBLEMAS.

LENGUAJES COMO PASCAL, COBOL, BASIC Y PL.I SON CONSIDERADOS COMO ORIENTADOS A PROCEDIMIENTOS; LENGUAJES COMO GPSS Y SPSS SON CONSIDERADOS COMO ORIENTADOS A PROBLEMAS.

2.26 COMPILADORES QUICK-AND-DIRTY. LOS COMPILADORES PARA OPTIMIZACION.

EN AMBIENTES DE DESARROLLO DE PROGRAMAS, LAS COMPILACIONES SON EFECTUADAS FRECUENTEMENTE Y LOS PROGRAMAS SON CORRIDOS CONCISAMENTE HASTA QUE UNA BASURA APARECE. PARA ESTO SE USAN LOS COMPILADORES QUICK-AND-DIRTY, LOS CUALES PRODUCEN UN CODIGO OBJETO RAPIDAMENTE, PERO ESTE PUEDE SER MUY INEFICIENTE EN TERMINOS DE CONSUMO DE MEMORIA Y VELOCIDAD DE EJECUCION. UNA VEZ QUE UN PROGRAMA HA SIDO DEPURADO Y ESTA LISTO PARA SER PUESTO EN PRODUCCION, UN COMPILADOR OPTIMIZANTE ES USADO PARA PRODUCIR CODIGO ALTAMENTE EFICIENTE. ESTE COMPILADOR CORRE MAS LENTO PERO LA CALIDAD DEL CODIGO OBJETO QUE PRODUCE ES MUY ALTA.

EN LOS 70 S SE CREIA QUE UN BUEN PROGRAMADOR EN LENGUAJE DE ENSAMBLADO PODIA PRODUCIR MEJOR CODIGO QUE UN COMPILADOR OPTIMIZANTE. LOS COMPILADORES DE HOY SON TAN EFECTIVOS QUE EL CODIGO QUE GENERAN IGUALA O EXCEDE LA CALIDAD DEL CODIGO PRODUCIDO POR UN PROGRAMADOR EN LENGUAJE DE ENSAMBLADO CON MUY ALTAS HABILIDADES.

LOS PROGRAMAS QUE NECESITAN SER MUY EFICIENTES (TALES COMO LOS S.O.) YA NO TIENEN QUE SER ESCRITOS EN LENGUAJES DE ENSAMBLADO. LA MAYORIA DE LOS S.O. DE HOY SON ESCRITOS EN LENGUAJES DE ALTO NIVEL Y TRADUCIDOS POR COMPILADORES OPTIMIZANTES DE MUY ALTA CALIDAD EN CODIGO DE MAQUINA EFICIENTE.

2.27 INTERPRETADORES.

UNA FORMA POPULAR E INTERESANTE DE TRADUCTOR, ES UN INTERPRETADOR QUE NO PRODUCE PROGRAMA OBJETO, ESTE CORRE UN PROGRAMA FUENTE DIRECTAMENTE. LOS INTERPRETES SON MUY POPULARES EN AMBIENTES DE DESARROLLO DE PROGRAMAS EN LOS CUALES LOS PROGRAMAS CORREN HASTA QUE UN ERROR ES ENCONTRADO. ESTOS TRADUCTORES EVITAN EL OVERHEAD DE ENSAMBLE O COMPILACION. TAMBIEN SON POPULARES EN COMPUTADORAS PERSONALES PERO LOS INTERPRETES CORREN LENTAMENTE COMPARADOS

CON CODIGO COMPILADO; DEBEN TRADUCIR CADA INSTRUCCION CADA VEZ QUE ES EJECUTADA.

2.28 CARGADORES ABSOLUTOS Y RELOCALIZABLES.

LOS PROGRAMAS DEBEN SER COLOCADOS EN MEMORIA PRINCIPAL PARA SER EJECUTADOS. LA ASOCIACION DE INSTRUCCIONES Y DATOS CON LOCALIZACIONES EN MEMORIA PRINCIPAL ES UNA TAREA ENORMEMENTE IMPORTANTE. EL TRABAJO DE HACER ESTA ASOCIACION ES ALGUNAS VECES DEJADA AL USUARIO, OTRAS AL TRADUCTOR, OTRAS A UN PROGRAMA DEL SISTEMA LLAMADO CARGADOR Y OTRAS AL S.O. LA ASOCIACION DE INSTRUCCIONES Y DATOS CON LOCALIZACIONES PARTICULARES DE MEMORIA ES LLAMADO VINCULOS (BINDING). AL PROGRAMAR EN LENGUAJE DE MAQUINA, EL BINDING (O VINCULOS) ES HECHO EN EL MOMENTO DE LA CODIFICACION.

LA TENDENCIA HA SIDO DIFERIR EL BINDING LO MAS POSIBLE. LOS SISTEMAS VIRTUALES HACEN EL BINDING DINAMICAMENTE CUANDO EL PROGRAMA SE EJECUTA. ASOCIADO CON EL RETRASO DE BINDING HAY UN INCREMENTO EN LA FLEXIBILIDAD PARA EL USUARIO Y EL SISTEMA. PERO EL COSTO ES TRADUCTORES MUY SOFISTICADOS, CARGADORES, HARDWARE Y SISTEMAS OPERATIVOS.

UN CARGADOR ES UN PROGRAMA QUE COLOCA INSTRUCCIONES Y DATOS DE PROGRAMA EN MEMORIA PRINCIPAL. UN CARGADOR ABSOLUTO COLOCA ESOS CAMPOS EN LAS LOCALIZACIONES INDICADAS POR EL PROGRAMA EN LENGUAJE DE MAQUINA.
GRANDES BIBLIOTECAS DE SUBRUTINAS SON PROPORCIONADAS DE TAL MANERA QUE SI UN PROGRAMADOR DESEA HACER CIERTAS OPERACIONES COMUNES, PUEDE USAR RUTINAS PROPORCIONADAS POR EL SISTEMA. I/O EN PARTICULAR ES NORMALMENTE MANEJADA POR RUTINAS EXTERNAS AL PROGRAMA DEL USUARIO.
ENTONCES, EL PROGRAMA EN LENGUAJE DE MAQUINA PRODUCIDO POR UN TRADUCTOR DEBE NORMALMENTE SER COMBINADO CON OTROS PROGRAMAS EN LENGUAJE MAQUINAL PARA FORMAR UNA UNIDAD DE EJECUCION UTIL. ESTE PROCESO ES EFECTUADO POR LOS CARGADORES ENCADENANTES (LINKING LOADERS) Y EDITORES ENCADENANTES (LINKAGE EDITORS) ANTES DE LA EJECUCION.

A TIEMPO DE CARGA, UN LINKING LOADER COMBINA LOS PROGRAMAS QUE SON REQUERIDOS Y LOS CARGA DIRECTAMENTE A MEMORIA. EL LINKAGE EDITOR TAMBIEN HACE ESTA COMBINACION DE PROGRAMAS. PERO CREA UNA IMAGEN DE CARGA QUE GUARDA EN ALMACENAMIENTO SECUNDARIO PARA REFERENCIAS FUTURAS.

EL LINKAGE EDITOR ES MUY UTIL EN AMBIENTES DE PRODUCCION. CUANDO UN PROGRAMA VA A SER EJECUTADO, LA IMAGEN DE CARGA PRODUCIDA POR EL LINKAGE EDITOR PUEDE SER CARGADA INMEDIATAMENTE SIN EL OVERHEAD (FRECUENTEMENTE SIGNIFICATIVO) DE RECOMBINAR PIEZAS DE PROGRAMA.

2.29 FIRMWARE.

EL CONCEPTO DE MICROPROGRAMACION SE APLICA AL PROCESADOR

MAURICE WILKES, EN SUS APUNTES DE 1951 PRESENTO LOS CONCEPTOS QUE FORMAN LAS TECNICAS DE MICROPROGRAMACION ACTUAL. SIN EMBARGO, NO FUE HASTA QUE APARECIO EL SISTEMA 360 EN LA MITAD DE LOS 60'S QUE LA MICROPROGRAMACION SE USO EN GRAN ESCALA. DURANTE LOS 60'S LAS MANUFACTURAS DE COMPUTADORAS USARON MICROPROGRAMACION PARA IMPLEMENTAR CONJUNTOS DE INSTRUCCIONES EN LENGUAJE DE MAQUINA.

LA MICROPROGRAMACION DINAMICA APARECIO A FINALES DE LOS 60'S Y PRINCIPIOS DE LOS 70'S. PERMITIO QUE NUEVOS MICROPROGRAMAS FUERAN CARGADOS FACILMENTE A MEMORIA DE CONTROL DESDE LA CUAL LOS PROGRAMAS SON EJECUTADOS. ENTONCES LOS CONJUNTOS DE INTRUCCIONES DE MAQUINA PUEDEN SER ALTERADOS DINAMICAMENTE Y FRECUENTEMENTE. NO ES INCONCEBIBLE QUE FUTUROS SISTEMAS DE MULTIPROGRAMACION PUEDAN PERMITIR A LOS USUARIOS DIFERENTES CONJUNTOS DE INSTRUCCIONES, Y QUE UNA PARTE DEL SWITCHEO DEL PROCESADOR ENTRE PROGRAMAS PODRIA INVOLUCRAR TAMBIEN EL SWITCHEO DEL CONJUNTO DE INSTRUCCIONES.

LA MICROPROGRAMACION INTRODUCE UNA CAPA DE PROGRAMACION BAJO EL LENGUAJE DE MAQUINA DE UNA COMPUTADORA. COMO TAL, HACE POSIBLE DEFINIR INSTRUCCIONES EN LENGUAJE DE MAQUINA. ESTO ES INTEGRAL DE LAS ARQUITECTURAS DE LAS COMPUTADORAS MODERNAS Y TIENE ENORME SIGNIFICACION LA EJECUCION (PERFORMANCE) DE LOS S.O. Y EN CONSIDERACIONES DE SEGURIDAD.

LOS MICROPROGRAMAS SON CORRIDOS EN UNA MEMORIA DE CONTROL DE MUY ALTA VELOCIDAD. ESTAN FORMADAS POR MICROINSTRUCCIONES INDIVIDUALES QUE SON MUCHO MAS ELEMENTALES EN NATURALEZA Y SENCILLAS EN FUNCION QUE LAS INSTRUCCIONES EN LENGUAJE DE MAQUINA CONVENCIONALES. EN SISTEMAS EN EL CUAL EL CONJUNTO DE INSTRUCCIONES EN LENGUAJE DE MAQUINA ES IMPLANTADA POR MICROPROGRAMACION, CADA INSTRUCCION ES IMPLEMENTADA POR UN PROGRAMA COMPLETO Y POSIBLEMENTE GRANDE. ESTO INMEDIATAMENTE IMPLICA QUE PARA QUE LA MICROPROGRAMACION SEA UTIL ES NECESARIO QUE LA MEMORIA DE CONTROL SEA MUCHO MAS RAPIDA QUE LA MEMORIA PRIMARIA.

2.30 MICROCODIGO HORIZONTAL Y VERTICAL.

LAS INSTRUCCIONES EN MICROCODIGO PUEDEN SER CLASIFICADAS EN VERTICALES Y HORIZONTALES.

LA EJECUCION VERTICAL DE MICROINSTRUCCIONES ES COMO LA EJECUCION DE INSTRUCCIONES EN LENGUAJE DE MAQUINA. UNA INSTRUCCION VERTICAL TIPICA ESPECIFICA EL MOVIMIENTO DE UNO (O UNOS CUANTOS) CAMPOS DE DATOS ENTRE REGISTROS.

EL MICROCODIGO HORIZONTAL ES DIFERENTE, CADA INSTRUCCION REQUIERE MUCHOS MAS BITS PUESTO QUE PUEDE ESPECIFICAR LA OPERACION PARALELA DE MOVIMIENTOS DE DATOS ENTRE MUCHOS O TODOS LOS REGISTROS DE DATOS EN LA UNIDAD DE CONTROL.

LAS MICROINSTRUCCIONES HORIZONTALES SON MUCHO MAS POTENTES

QUE LAS VERTICALES. PERO LOS PROGRAMAS RESULTANTES PUEDEN SER MAS DIFICILES DE CODIFICAR Y DEPURAR.

2.31 DECIDIR QUE FUNCIONES IMPLEMENTAR EN MICROCODIGO.

ESTA ES UNA DESICION IMPORTANTE DE DISENO .
EL MICROCODIGO PRESENTA UNA REAL OPORTUNIDAD DE MEJORAR LA EJECUCION (PERFORMANCE) DE UN PROGRAMA.

2.32 EMULACION.

EMULACION ES UNA TECNICA EN LA CUAL UNA MAQUINA SE VE COMO SI FUERA OTRA. EL CONJUNTO DE INSTRUCCIONES DE LA MAQUINA A SER EMULADA ES MICROPROGRAMADA EN LA MAQUINA ANFITRION (HOST). LOS PROGRAMAS EN LENGUAJE MAQUINAL DE LA MAQUINA EMULADA PUEDEN SER CORRIDOS DIRECTAMENTE EN EL HOST. LOS VENDEDORES COMPUTACIONALES HACEN USO EXTENSIVO DE LA EMULACION CUANDO INTRODUCEN NUEVOS SISTEMAS. LOS USUARIOS CONSIGNADOS A COMPUTADORAS ANTIGUAS PUEDEN CORRER SUS PROGRAMAS DIRECTAMENTE EN LAS NUEVAS COMPUTADORAS SIN ALTERACION. ESTO SUAVIZA EL PROCESO DE CONVERSION.

2.33 MICRODIAGNOSTICO.

LOS MICROPROGRAMAS TIENEN MAS ACCESO QUE LOS PROGRAMAS EN LENGUAJE DE MAQUINA.
ES POSIBLE HACER MAS EXTENSA LA DETECCION Y CORRECCION DE ERRORES Y HACER ESAS OPERACIONES EN UN NIVEL MAS FINO. ALGUNOS SISTEMAS INTERCALAN MICRODIAGNOSTICO CON INSTRUCCIONES EN LENGUAJE MAQUINAL. ESTO HACE POSIBLE EVITAR PROBLEMAS POTENCIALES Y TENER OPERACION MAS CONFIABLE.

2.34 COMPUTADORAS PERSONALIZADAS.

DEBIDO A LO CARO DE DISENAR, CONSTRUIR Y MERCADEAR UN SISTEMA COMPUTACIONAL, LOS VENDEDORES SE HAN CONCENTRADO EN PRODUCIR MAQUINAS DE PROPOSITO GENERAL. LA GRAN INVERSION NECESARIA PARA PRODUCIR UN NUEVO SISTEMA HACE NECESARIAS GRANDES VENTAS PARA RECUPERAR COSTOS Y OBTENER UTILIDAD. LOS VENDEDORES HAN TENDIDO A EVITAR CONSTRUIR SISTEMAS DE PROPOSITO ESPECIAL, O DE UN SOLO TIPO. ESTO SE HA DEJADO A LAS UNIVERSIDADES EN LAS CUALES TALES SISTEMAS SE HAN CONSTRUIDO PRIMARIAMENTE POR SU VALOR EN INVESTIGACION.
LOS USUARIOS DE COMPUTADORAS SE HAN ENFRENTADO CON LA TAREA DE ADAPTAR COMPUTADORAS A SUS NECESIDADES. ESTA ADAPTACION SE HA HECHO POR SOFTWARE. EL HARDWARE PROVEE UN AMBIENTE DE PROPOSITO GENERAL PARA CORRER PROGRAMAS DE

SOFTWARE.
EN ALGUNOS SISTEMAS, LOS USUARIOS PUEDEN HACER ESTA ADAPTACION POR MICROCODIGO; PUEDEN USAR EL QUE LES PROPORCIONA EL VENDEDOR O ESCRIBIR EL SUYO.

2.35 AYUDA DEL MICROCODIGO.

LOS VENDEDORES CON FRECUENCIA PROPORCIONAN OPCIONES DE MEJORAMIENTO DE EJECUCION (PERFORMANCE) EN MICROCODIGO. IBM HA HECHO ESTO SATISFACTORIAMENTE CON SU SISTEMA OPERATIVO VM. ESTE SISTEMA IMPLEMENTA MULTIPLES MAQUINAS VIRTUALES PARA UTILIZACION CUIDADOSA DEL MECANISMO DE INTERRUPCION. LA AYUDA DE MICROCODIGO IMPLANTA UN NUMERO DE LAS RUTINAS DE MANEJO DE INTERRUPCIONES MAS FRECUENTEMENTE EJECUTADAS EN MICROCODIGO PARA LOGRAR MEJORAMIENTOS DE PERFORMANCE SIGNIFICANTES.

2.36 LA MICROPROGRAMACION Y S.O.

HAY PORCIONES DEL S.O. QUE ESTAN ENTRE LAS SECUENCIAS DE INSTRUCCIONES MAS FRECUENTEMENTE EJECUTADAS. EN UN SISTEMA INTERACTIVO DE PROCESAMIENTO DE TRANSACCIONES, POR EJEMPLO, EL MECANISMO DE DESPACHO QUE SELECCIONA LA SIGUIENTE UNIDAD DE TRABAJO A LA CUAL EL PROCESADOR SERA ASIGNADO PODRIA SER EJECUTADO CIENTOS DE VECES POR SEGUNDO. EL MECANISMO DE DESPACHO DEBE EJECUTARSE EFICIENTEMENTE; EL COLOCARLO EN MICROCODIGO ES UNA MANERA DE HACERLO MAS RAPIDO.

ALGUNAS FUNCIONES IMPLEMENTADAS EN MICROCODIGO:

- MANEJO DE INTERRUPCIONES (INTERRUPTS)
- MANTENIMIENTO DE VARIOS TIPOS DE ESTRUCTURAS DE DATOS
- PRIMITIVOS DE SINCRONIZACION
- OPERACIONES PARCIALES SOBRE PALABRAS
- CONMUTACION DE CONTEXTOS
- SECUENCIAS DE LLAMADA Y RETORNO A PROCEDIMIENTOS.

LA IMPLEMENTACION DE FUNCIONES DEL SISTEMA OPERATIVO EN MICROCODIGO PUEDE MEJORAR LA EJECUCION (PERFORMANCE), REDUCIR COSTOS DE DESARROLLO DE PROGRAMAS Y MEJORAR LA SEGURIDAD DEL SISTEMA.

ADMINISTRACION DE PROCESOS

3.1 INTRODUCCION.

AQUI NOSOTROS INTRODUCIMOS LA NOCION DE PROCESO. QUE ES CENTRAL PARA EL ENTENDIMIENTO DE LOS SISTEMAS MULTIUSUARIOS DE HOY EN DIA. ALGUNAS DE LAS DEFINICIONES MAS POPULARES SON PRESENTADAS, PERO AUN NO SE HA ENCONTRADO ALGUNA DEFINICION PERFECTA DE LO QUE PROCESO SIGNIFICA EN LA LITERATURA.
EL CONCEPTO DE ESTADOS DE PROCESO DIRECTO ES EXPLICADO, COMO TAMBIEN UNA DISCUSION ACERCA DE COMO LOS PROCESOS HACEN TRANSICIONES ENTRE LOS ESTADOS. TAMBIEN ES CONSIDERADO UN NUMERO BASICO DE OPERACIONES QUE PUEDEN SER EJECUTADAS DESDE LOS PROCESOS.
LAS DEFINICIONES Y CONCEPTOS PRESENTADOS AQUI SIRVEN COMO BASE PARA LAS DISCUCIONES DE PROCESOS CONCURRENTES ASINCRONICOS Y PROCESOS SCHEDULING DISCUTIDOS POSTERIORES.

3.2 DEFINICIONES DE PROCESO.

EL TERMINO PROCESO FUE USADO PRIMERO POR LOS DISENADORES DEL SISTEMA MULTICS EN LOS 60'S. DESDE ENTONCES, PROCESO LLAMADO ALGUNAS VECES TASK, HA TENIDO MUCHAS DEFINICIONES.
ALGUNAS DE ELLAS SON:

- UN PROGRAMA EN EJECUCION
- UNA ACTIVIDAD ASINCRONICA
- EL ESPIRITU ANIMADO DE UN PROCEDIMIENTO
- EL PUNTO DE CONTROL DE UN PROCEDIMIENTO EN EJECUCION EL CUAL ES MANIFESTADO POR LA EXISTENCIA DE UN BLOCK DE CONTROL DE PROCESO EN EL SISTEMA OPERATIVO
- AQUELLA ENTIDAD PARA LA CUAL SON ASIGNADOS LOS PROCESADORES
- LA UNIDAD DESPACHABLE

3.3 ESTADOS DEL PROCESO.

¿ Como se define el estado discreto de un procesador.

DURANTE SU EXISTENCIA, UN PROCESO VA ATRAVES DE UNA SERIE DE ESTADOS DISCRETOS VARIOS EVENTOS PUEDEN HACER QUE UN PROCESO CAMBIE DE ESTADO.
SE DICE QUE UN PROCESO ESTA CORRIENDO (EN EL ESTADO DE EJECUCION) SI ESTE TIENE ACTUALMENTE AL CPU.
SE DICE QUE ESTA DISPONIBLE (EN EL ESTADO DISPONIBLE) SI ESTE PUDIERA USAR UN CPU SI ALGUNO ESTUVIERA DISPONIBLE.
SE DICE QUE UN PROCESO ESTA BLOQUEADO (EN ESTADO BLOQUEADO) SI ESTE ESTA ESPERANDO POR ALGUN EVENTO A OCURRIR (COMO LA TERMINACION DE UNA OPERACION DE I/O, POR EJEMPLO) ANTES DE QUE ESTA PUEDA PROSEGUIR.
EXISTEN OTROS ESTADOS DE PROCESO PERO POR EL MOMENTO LA

DISCUSION SE CONCENTRARA EN ESTOS 3 ESTADOS.
PARA SIMPLIFICAR, CONSIDEREMOS UN SISTEMA DE UN SOLO CPU, SIN EMBARGO EL EXTENDERSE EL MULTIPROCESAMIENTO NO ES PROBLEMATICO. SOLAMENTE UN PROCESO PUEDE ESTAR CORRIENDO A UN TIEMPO PERO VARIOS PROCESOS PUEDEN ESTAR DISPONIBLES Y VARIOS PUEDEN ESTAR BLOQUEADOS DE ESTA MANERA ESTABLECEMOS UNA LISTA DE DISPONIBLES PARA PROCESOS DISPONIBLES Y UNA LISTA DE BLOQUEADOS PARA PROCESOS BLOQUEADOS. LA LISTA DE DISPONIBLES SE MANTIENE EN ORDEN PRIORITARIO DE MANERA QUE EL SIGUIENTE PROCESO EN RECIBIR AL CPU ES EL PRIMER PROCESO EN LA LISTA. LA LISTA DE BLOQUEADOS NO TIENE ORDEN ALGUNO YA QUE LOS PROCESOS NO SE CONVIERTEN EN DESBLOQUEADOS (ESTO ES DISPONIBLES) EN ORDEN PRIORITARIO; EN VEZ DE ESTO SE DESBLOQUEAN Y TAL VEZ ESTOS NUNCA LLEGUEN A OCURRIR EN EL ORDEN EN EL CUAL OCURRAN LOS EVENTOS QUE ELLOS ESPEREN.

3.4 ¿TRANSICIONES ENTRE LOS ESTADOS DE LOS PROCESOS?

CUANDO UN JOB ES ADMITIDO EN EL SISTEMA, UN PROCESO CORRESPONDIENTE ES CREADO Y ENTONCES ES INCERTADO AL FINAL DE LA LISTA DE DISPONIBLES. EL PROCESO SE MUEVE GRADUALMENTE HACIA LA CABEZA DE LA LISTA CONFORME LOS PROCESOS ANTERIORES SE VAYAN COMPLETANDO. CUANDO EL PROCESO ALCANCE LA CABEZA DE LA LISTA Y CUANDO EL CPU ESTE DISPONIBLE, EL PROCESO TOMA AL CPU Y SE DICE QUE HIZO UNA TRANSICION DE ESTADO, DEL ESTADO DISPONIBLE AL ESTADO DE EJECUCION. LA ASIGNACION DEL CPU AL PRIMER PROCESO EN LA LISTA DE DISPONIBLES ES LLAMADO DESPACHAR Y EJECUTADO POR UNA ENTIDAD DEL SISTEMA LLAMADA DESPACHADOR.
INDICAREMOS ESTA TRANSICION DE LA SIGUIENTE MANERA:

DISPATCH(PROCESSNAME):READY --> RUNNING

MIENTRAS QUE EL PROCESO TENGA AL CPU, SE DICE QUE ESTA CORRIENDO. PARA PREEVER QUE ALGUN PROCESO MONOPOLIZE EL SISTEMA YA SEA ACCIDENTAL O APROPOSITO, EL S.O. ESTABLECE UN RELOJ INTERRUPTOR DE HARDWARE PARA PERMITIR A ESTE USUARIO CORRER POR UN INTERVALO DE TIEMPO ESPECIFICO O QUANTUM. SI EL PROCESO NO LIBERA AL CPU VOLUNTARIAMENTE ANTES DE QUE EL INTERVALO EXPIRE, EL RELOJ GENERA UNA INTERRUPCION CAUSANDO QUE EL S.O. RETOME EL CONTROL. ENTONCES, EL SISTEMA OPERATIVO HACE DISPONIBLE AL PROCESO QUE ESTABA CORRIENDO ANTERIORMENTE, Y HACE CORRE EL PRIMER PROCESO QUE ESTABA EN LISTA DE DISPONIBLES. ESTAS TRASICIONES DE ESTADO SE INDICAN COMO:

TIMERRUNNOUT(PROCESSNAME): RUNNING --> READY
Y DISPATCH(PROCESSNAME): READY --> RUNNING

SI UN PROCESO EN EJECUCION INDICA UNA OPERACION DE I/O ANTES

DE QUE SU QUANTUM EXPIRE, EL PROCESO EN EJECUCION LIBERA VOLUNTARIAMENTE AL CPU (EL PROCESO SE DESBLOQUEA A SI MISMO HASTA LA TERMINACION DE LA OPERACION DE I/O). ESTA TRANSICION DE ESTADO ES:

BLOCK(PROCESSNAME): RUNNING --> BLOCKED

LA OTRA UNICA TRANSICION DE ESTADO PERMITIDA EN NUESTRO MODELO DE 3 ESTADOS OCURRE CUANDO UNA OPERACION DE I/O (O ALGUNOS OTROS PROCESOS POR LOS CUALES EL PROCESO ESPERE) TERMINE. EL PROCESO HACE LA TRANSICION DEL ESTADO BLOQUEADO AL ESTADO DISPONIBLE. LA TRANSICION ES:

WAKEUP(PROCESSNAME): BLOCKED --> READY

ENTONCES HEMOS DEFINIDO 4 POSIBLES TRANSICIONES DE ESTADO:

DISPATCH(PROCESSNAME): READY --> RUNNING
TIMERRUNOUT(PROCESSNAME): RUNNING --> READY
BLOCK(PROCESSNAME): RUNNING --> BLOCKED
WAKEUP(PROCESSNAME): BLOCKED --> READY

NOTESE QUE LA UNICA TRANSICION DE ESTADO INICIADA POR EL PROPIO PROCESO USUARIO ES EL BLOCK Y LAS OTRAS 3 TRANSICIONES SON INICIADAS POR ENTIDADES EXTERNAS A EL PROCESO.

3.5 EL BLOCK DE CONTROL DEL PROCESO (PCB)

LA MANIFESTACION DE UN PROCESO EN UN SISTEMA OPERATIVO ES UN BLOCK DE CONTROL DE PROCESO. EL PCB ES UNA ESTRUCTURA DE DATOS QUE CONTIENE CIERTA INFORMACION IMPORTANTE ACERCA DEL PROCESO Y QUE INCLUYE:

- EL ESTADO ACTUAL DEL PROCESO
- IDENTIFICACION UNICA DEL PROCESO
- LA PRIORIDAD DEL PROCESO
- APUNTADORES PARA LOCALIZAR LA MEMORIA DEL PROCESO
- APUNTADORES PARA ALOJAR RECURSOS
- UNA AREA "SAVE" DE REGISTRO

EL PCB ES UN ALMACEN CENTRAL DE INFORMACION QUE PERMITE AL S.O. TENER TODA LA INFORMACION MAESTRA ACERCA DEL PROCESO. CUANDO EL S.O. CAMBIA LA ATENCION DEL CPU ENTRE LOS PROCESOS, ESTE USA LAS AREAS DE "SAVE" EN EL PCB PARA RETENER LA INFORMACION QUE NECESITA PARA REINICIAR CADA PROCESO CUANDO EL SIGUIENTE PROCESO TENGA AL CPU. ENTONCES EL PCB ES LA ENTIDAD QUE DEFINE UN PROCESO PARA EL S.O. DEBIDO A QUE LOS PCB NECESITAN SER MANIPULADOS RAPIDAMENTE POR EL S.O. MUCHOS SISTEMAS COMPUTACIONALES CONTIENEN UN REGISTRO DE HARDWARE QUE SIEMPRE APUNTA AL PCB DEL PROCESO

QUE SE ESTA EJECUTANDO ACTUALMENTE. EXISTEN INSTRUCCIONES EN HARDWARE CASI SIEMPRE DISPONIBLES QUE CARGAN INFORMACION DEL ESTADO HACIA EL PCB Y REINCORPORAN LA INFORMACION RAPIDAMENTE.

3.6 OPERACIONES EN LOS PROCESOS.

LOS SISTEMAS QUE MANEJAN A LOS PROCESOS DEBEN SER CAPACES DE EJECUTAR CIERTAS OPERACIONES EN LOS PROCESOS, ESTAS INCLUYEN:

- CREAR UN PROCESO
- DESTRUIR UN PROCESO
- SUSPENDER UN PROCESO
- CONCLUIR UN PROCESO
- CAMBIAR LA PRIORIDAD DE UN PROCESO
- BLOQUEAR UN PROCESO
- DESPERTAR UN PROCESO
- DEPACHAR UN PROCESO

EL CREAR UN PROCESO INVOLUCRA MUCHAS OPERACIONES COMO:

- NOMBRAR EL PROCESO
- INSERTARLO EN LA LISTA DE PROCESOS DEL SISTEMA
- DETERMINAR LA PRIORIDAD INICIAL DEL PROCESO EN EL SISTEMA
- CREAR EL PCB
- ALOJAR LOS RECURSOS INICIALES DEL PROCESO

UN PROCESO PUEDE CREAR UN NUEVO PROCESO, SI ESTO OCURRE EL PROCESO CRADOR ES LLAMADO PROCESO PADRE Y EL PROCESO CREADO ES LLAMADO PROCESO HIJO. DICHA CREACION PERMITE UNA ESTRUCTURA DE PROCESO JERARQUICA, EN LA CUAL CADA HIJO TIENE SOLAMENTE UN PADRE, PERO CADA PADRE PUEDE TENER MUCHOS HIJOS.

EL DESTRUIR UN PROCESO INVOLUCRA DESLIGARLO DEL SISTEMA Y SUS RECURSOS SE DEVUELVEN AL SISTEMA, ESTE ES PURGADO DE CUALQUIER LISTA O TABLA DEL SISTEMA, Y SU BLOCK DE CONTROL DE PROCESO ES BORRADO.

UN PROCESO SUSPENDIDO NO PUEDE PROSEGUIR HASTA QUE OTRO LO CONCLUYA. EL ESTAR SUSPENDIDO ES UNA OPERACION MUY IMPORTANTE Y HA SIDO IMPLEMENTADA EN UNA VARIEDAD DE FORMAS EN MUCHOS SISTEMAS DIFERENTES.

LAS SUSPENCIONES NORMALMENTE DURAN SOLO PERIODOS CORTOS DE TIEMPO, GENERALMENTE SON EFECTUADAS POR EL SISTEMA PARA QUITAR CIERTOS PROCESOS TEMPORALMENTE DURANTE UNA SITUACION PICO DE CARGA. PARA SUSPENCIONES A LARGO PLAZO LOS RECURSOS DEL PROCESO DEBERAN SER LIBERADOS. LA DESICION DE LIBERAR RECURSOS DEPENDE MUCHO DE LA NATURALEZA DEL RECURSO. MEMORIA PRINCIPAL DEBERA SER LIBERADA INMEDIATAMENTE CUANDO UN PROCESO ES SUSPENDIDO. UN DRIVE DE CINTA PUEDE SER RETENIDO POR UN PROCESO SUSPENDIDO SOLO MOMENTANEAMENTE, PERO DEBERA SER LIBERADO POR UN PROCESO SUSPENDIDO POR UN PERIODO LARGO

O INDEFINIDO.
RESUMIENDO (O ACTIVANDO) UN PROCESO INVOLUCRA RECOMENZARLO DESDE EL PUNTO EN QUE FUE SUSPENDIDO.
LA DESTRUCCION DE UN PROCESO ES MAS COMPLICADA CUANDO EL PROCESO HA GENERADO OTROS PROCESOS. EN ALGUNOS SISTEMAS UN PROCESO GENERADO ES DESTRUIDO AUTOMATICAMENTE CUANDO SU PADRE ES DESTRUIDO; EN OTROS SISTEMAS LOS PROCESOS GENERADOS PROCEDEN INDEPENDIENTEMENTE DE SUS PADRES Y LA DESTRUCCION DE UN PADRE NO TIENE NINGUN EFECTO SOBRE LOS HIJOS DEL PADRE DESTRUIDO.
EL CAMBIAR LA PRIORIDAD DE UN PROCESO NORMALMENTE INVOLUCRA NADA MAS QUE MODIFICAR EL VALOR DE LA PRIORIDAD EN EL PCB.

¿ como TRABAJO

3.7 EL SUSPEND Y RESUME.?

EN LA SECCION PREVIA LAS NOCIONES DE SUSPENDER Y RESUMIR PROCESOS SE INTRODUJERON. ESTAS SON OPERACIONES IMPORTANTES POR VARIAS RAZONES:

- SI UN SISTEMA ESTA FUNCIONANDO POBREMENTE Y PUEDE FALLAR, ENTONCES LOS PROCESOS ACTUALES DEBEN SER SUSPENDIDOS PARA SER RESUMIDOS DESPUES DE QUE EL PROBLEMA ES CORREGIDO.
- UN USUARIO QUE SOSPECHE ACERCA DE LOS RESULTADOS PARCIALES DE UN PROCESO DEBE SUSPENDERLO (MAS BIEN QUE ABORTARLO) HASTA QUE EL USUARIO PUEDE AFIRMAR QUE EL PROCESO FUNCIONA O NO CORRECTAMENTE.
- EN RESPUESTA A FLUCTUACIONES CORTAS EN LA CARGA DEL SISTEMA, ALGUNOS PROCESOS PUEDEN SER SUSPENDIDOS Y RESUMIDOS MAS TARDE CUANDO LA CARGA VUELVE A SUS NIVELES NORMALES.

DOS NUEVOS ESTADOS HAN SIDO AGREGADOS, SUSPENDEDREADY Y SUSPENDEDBLOCKED; NO HAY NECESIDAD FARA UN ESTADO SUSPENDEDRUNNING. UNA SUSPENSION PUEDE SER INICIADA POR EL PROCESO MISMO O POR OTRO PROCESO.
EN UN SISTEMA UNIPROCESADOR UN PROCESO EN EJECUCION SE PUEDE SUSPENDER A SI MISMO; NINGUN OTRO PROCESO PUEDE ESTAR CORRIENDO AL MISMO TIEMPO QUE ENVIE EL SUSPEND. EN UN SISTEMA MULTIPROCESADOR, UN PROCESO CORRIENDO PUEDE SER SUSPENDIDO POR OTRO PROCESO CORRIENDO EN UN PROCESADOR DIFERENTE.
UN PROCESO READY PUEDE SER SUSPENDIDO SOLO POR OTRO PROCESO. HACE LA SIGUIENTE TRANSICION:

SUSPEND(PROCESSNAME):READY-->SUSPENDEDREADY.

UN PROCESO SUSPENDEDREADY PUEDE SER PUESTO READY POR OTRO PROCESO. HACE LA TRANSICION:

RESUME(PROCESSNAME):SUSPENDEDREADY-->READY.

UN PROCESO BLOQUEADO PUEDE SER SUSPENDIDO POR OTRO PROCESO. HACE LA SIGUIENTE TRANSICION:

SUSPEND(PROCESSNAME):BLOCKED-->SUSPENDEDBLOCKED.

UN PROCESO SUSPENDEDBLOCKED PUEDE SER RESUMIDO POR OTRO PROCESO. HACE LA SIGUIENTE TRANSICION:

RESUME(PROCESSNAME): SUSPENDEDBLOCKED-->BLOCKED.

UNO PODRIA ARGUMENTAR QUE EN LUGAR DE SUSPENDER UN PROCESO BLOCKEADO, ES MEJOR ESPERAR HASTA QUE SE COMPLETE EL I/O U OTRO EVENTO Y EL PROCESO ESTE READY. DESAFORTUNADAMENTE, LA TERMINACION PODRIA NO VENIR NUNCA, O PUEDE SER RETARDADA INDEFINIDAMENTE. ASI, EL DISENADOR SE ENFRENTA CON EFECTUAR LA SUSPENCION DEL PROCESO BLOQUEADO O ESTABLECER UN MECANISMO TAL QUE CUANDO LA TERMINACION OCURRA, LA SUSPENCION SERA HECHA DESDE EL ESTADO READY DEBIDO A QUE LA SUSPENCION ES CON FRECUENCIA UNA ACTIVIDAD DE ALTA PRIORIDAD, DEBERIA SER EFECTUADA INMEDIATAMENTE. CUANDO LA TERMINACION FINALMENTE OCURRE (SI ES QUE OCURRE), EL PROCESO SUSPENDEDBLOCKED HACE LA SIGUIENTE TRANSICION:

COMPLETION(PROCESSNAME):SUSPENDEDBLOCKED-->SUSPENDEDREADY.

EL USO DE SUSPEND Y RESUME POR EL S.O. PARA BALANCEAR LA CARGA DEL SISTEMA SE DISCUTE MAS TARDE.

3.8 PROCESAMIENTO DE INTERRUPCIONES (INTERRUPTS).

EN UN SISTEMA COMPUTACIONAL, UNA INTERRUPCION (INTERRUPT) ES UN EVENTO QUE ALTERA LA SECUENCIA EN LA CUAL UN PROCESADOR EJECUTA INSTRUCCIONES, ES GENERADA POR EL HARDWARE DEL SISTEMA COMPUTACIONAL.
CUANDO OCURRE UNA INTERRUPCION:
- EL S.O. OBTIENE EL CONTROL
- EL S.O. SALVA EL ESTADO DEL PROCESO INTERRUMPIDO. EN MUCHOS SISTEMAS ESTA INFORMACION ES ALMACENADA EN EL PCB DEL PROCESO INTERRUMPIDO.
- EL S.O. ANALIZA LA INTERRUPCION Y PASA EL CONTROL A LA RUTINA APROPIADA PARA MANEJAR LA INTERRUPCION.

UNA INTERRUPCION PUEDE SER ESPECIFICAMENTE INICIADA POR UN PROCESO CORRIENDO, O PUEDE SER CAUSADA POR ALGUN EVENTO QUE PUEDE O NO ESTAR RELACIONADO AL PROCESO EN EJECUCION.

3.9 TIPOS DE INTERRUPTS.

EN ESTA SECCION EL ESQUEMA DE INTERRUPCION DE LOS PROCESADORES DE GRAN TAMANO DE IBM ES DISCUTIDO.
ESTA INFORMACION ES ESPECIALMENTE UTIL EN EL TEXTO CUANDO LOS CASOS DE ESTUDIO DE LOS SISTEMAS OPERATIVOS MVS Y VM DE IBM SON PRESENTADOS.
HAY 6 TIPOS DE INTERRUPCION, ESTAS SON:
-SVC: SUPERVISOR DE LLAMADAS (SUPERVISOR CALL)

SON INICIADAS POR UN PROCESO CORRIENDO QUE EJECUTA LA INSTRUCCION SVC. UNA SVC ES UNA PETICION GENERADA POR EL USUARIO POR UN SISTEMA PARTICULAR DE SERVICIO TAL COMO EFECTUAR UN I/O, OBTENER MAS MEMORIA, O COMUNICACION CON EL OPERADOR DEL SISTEMA. EL MECANISMO SVC AYUDA A CUIDAR AL S.O. DE LOS USUARIOS. UN USUARIO NO PUEDE ENTRAR AL S.O. ARBITRARIAMENTE; EL USUARIO DEBE PEDIR UN SERVICIO ATRAVES DE UN SVC. EL S.O. ESTA CONCIENTE DE TODOS LOS INTENTOS DEL USUARIO DE CRUZAR SUS FRONTERAS, Y PUEDE RECHAZAR CIERTAS PETICIONES SI EL USUARIO NO TIENE LOS PRIVILEGIOS APROPIADOS

-INTERRUPCIONES DE ENTRADA/SALIDA (INTERRUPTS DE I/O)
SON INICIADAS POR EL HARDWARE DE I/O. SENALAN AL CPU QUE EL ESTADO DE UN CANAL O DISPOSITIVO HA CAMBIADO. LAS INTERRUPTS DE I/O SON CAUSADAS CUANDO UNA OPERACION DE I/O TERMINA, CUANDO UN ERROR OCURRE, O CUANDO UN DISPOSITIVO ES PUESTO EN READY.

-INTERRUPCIONES EXTERNAS (INTERRUPTS EXTERNAL)
SON CAUSADAS POR VARIOS EVENTOS INCLUYENDO LA EXPIRACION DE UN QUANTUM EN UN RELOJ, LA PRESION DE LA TECLA DE INTERRUPT DE LA CONSOLA POR EL OPERADOR, O LA RECEPCION DE UNA SENAL DESDE OTRO PROCESADOR EN UN SISTEMA MULTIPROCESADOR.

- INTERRUPCIONES DE REINICIO (INTERRUPTS RESTART)
SON CAUSADAS CUANDO EL OPERADOR PRESIONA EL BOTON DE REINICIO, O CUANDO UN SENAL DE REINICIO LLEGA DESDE OTRO PROCESADOR.

- INTERRUPCION POR CHEQUEO DE PROGRAMA (PROGRAM CHECK)
SON CAUSADAS POR VARIOS TIPOS DE ERRORES EXPERIMENTADOS POR UN PROCESO CORRIENDO TAL COMO UN INTENTO DE DIVIDIR POR CERO, INTENTO DE UN PROCESO DE USUARIO DE EJECUTAR UNA INSTRUCCION PRIVILEGIADA, INTENTO DE EJECUTAR UN CODIGO DE OPERACION INVALIDO, ETC.

- INTERRUPCION POR CHEQUEO DE MAQUINA (MACHINE CHECK)
SON CAUSADAS POR HARDWARE MAL FUNCIONANDO.

3.10 CONMUTACION DE CONTEXTO (CONTEXT SWITCHING)

EL SISTEMA OPERATIVO INCLUYE RUTINAS LLAMADAS MANEJADORES DE INTERRUPCIONES (INTERRUPTS HADLER'S (IH'S)) PARA PROCESAR CADA TIPO DE INTERRUPCION, HAY 6 TIPOS DE IH'S: EL SVC IH, EL I/O IH, EL EXTERNAL IH, EL RESTART IH, EL PROGRAM CHECK IH Y EL MACHINE CHECK IH.
CUANDO UNA INTERRUPCION OCURRE EL S.O. SALVA EL ESTADO DEL PROCESO INTERRUMPIDO Y RUTEA EL CONTROL AL MANEJADOR APROPIADO. ESTO SE HACE MEDIANTE LA TECNICA CONMUTACION DE CONTEXTO (CONTEXT SWITCHING).
LAS PSWs (PROGRAM STSTUS WORDS) CONTROLAN EL ORDEN DE EJECUCION DE INSTRUCCIONES Y CONTIENEN INFORMACION ACERCA DEL ESTADO DEL PROCESO.
HAY 3 TIPOS DE PSWs: PSWs ACTUAL, PSWs NUEVA Y PSWs VIEJAS.

LA DIRECCION DE LA SIGUIENTE INSTRUCCION A SER EJECUTADA ES GUARDADA EN EL PSW ACTUAL QUE TAMBIEN INDICA LOS TIPOS DE INTERRUPTS ACTUALMENTE HABILITADOS, Y LOS DESHABILITADOS PERMANECEN PENDIENTES, O IGNORADOS EN ALGUNOS CASOS. EL PROCESADOR NO DEBE NUNCA SER DESHABILITADO POR INTERRUPCIONES SVC, RESTART, O ALGUN TIPO DE INTERRUPTS DE PROGRAMA. LAS RAZONES PARA HABILITAR O DESHABILITAR INTERRUPTS SE VERAN CLARAMENTE MAS TARDE.

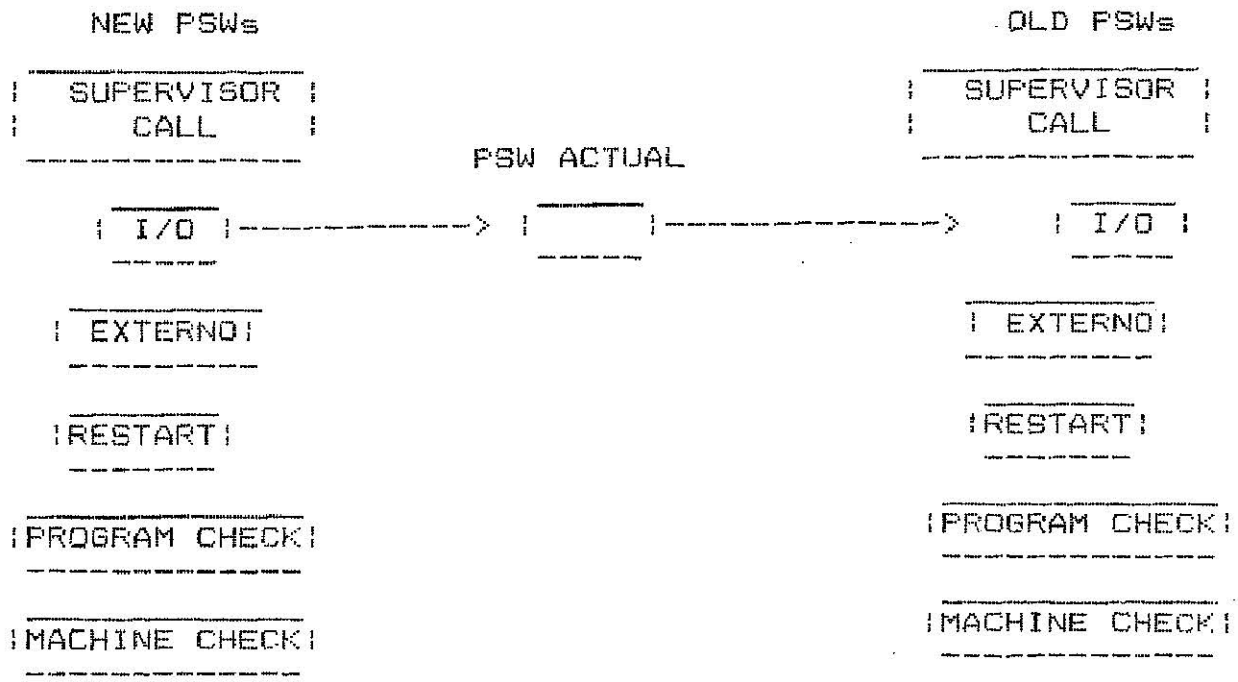
EN UN SISTEMA UNIPROCESADOR HAY SOLAMENTE LA PSW ACTUAL, PERO HAY 6 NUEVAS PSW (UNA PARA CADA INTERRUPT), Y 6 PSW ANTIGUAS (OLD) UNA PARA CADA TIPO DE INTERRUPCION.

LA NUEVA PSW PARA UN TIPO DE INTERRUPT DADO CONTIENE LA DIRECCION PERMANENTE DE HARDWARE EN LA CUAL EL MANEJADOR DE INTERRUPT PARA ESE TIPO DE INTERRUPT RESIDE.

CUANDO OCURRE UNA INTERRUPCION (VER LA SIGUIENTE FIGURA) SI EL PROCESADOR NO ES INHABILITADO PARA ESE TIPO DE INTERRUPCION, ENTONCES EL HARDWARE AUTOMATICAMENTE SWITCHEA LOS PSWs PARA ALMACENAR EL NUEVO PSW PARA ESE TIPO DE INTERRUPCION DENTRO DEL PSW ACTUAL.

DESPUES DE ESTE CAMBIO DE PSW EL ACTUAL PSW AHORA CONTIENE LA DIRECCION DEL APROPIADO INTERRUPTOR MANUAL. EL INTERRUPTOR MANUAL DESPUES PROCESA LA INTERRUPCION.

CUANDO EL PROCESO DE INTERRUPCION ESTE COMPLETO, EL CPU ES DESPACHADO PARA EL PROCESO QUE ESTABA CORRIENDO AL MOMENTO DE LA INTERRUPCION, PARA EL PROCESO QUE ESTE EN READY DE MAS ALTA PRIORIDAD. ESTO DEPENDE YA SEA QUE EL PROCESO DE INTERRUPCION SEA PRIVILEGIADO O NO PRIVILEGIADO. SI EL PROCESO ES NO PRIVILEGIADO, TOMA DE NUEVO AL CPU. SI EL PROCESO ES PRIVILEGIADO, ESTE TOMA AL CPU SOLO SI LOS PROCESOS SON READY.



EXISTEN MUCHOS ESQUEMAS DE INTERRUPCION SIGNIFICANTES MAS DE LOS QUE SE DESCRIBEN AQUI.

3.11 EL NUCLEO DEL SISTEMA OPERATIVO

TODAS LAS OPERACIONES ENVUELTAS EN UN PROCESO SON CONTROLADAS POR UNA PEQUENA PORCION DEL S.O. QUE CONTROLA OPERACIONES QUE INVOLUCRAN PROCESOS, EL NUCLEO REPRESENTA SOLO UNA PEQUENA PORCION DEL CODIGO DEL S.O.RUTINAS RESIDENTES SIEMPRE ESTAN CARGADAS EN MEMORIA Y SE LES LLAMA NUCLEO KERNEL O CORE.
EL NUCLEO ORDINARIAMENTE ESTA EN ALMACENAMIENTO PRIMARIO MIENTRAS QUE OTRAS PORCIONES DEL S.O. SON LANZADOS DONDE SON NECESARIOS EN ALMACENAMIENTO SECUNDARIO.
UNA DE LAS FUNCIONES MAS IMPORTANTES INCLUIDAS EN EL NUCLEO ES UN PROCESO INTERRUPTOR. A LA LARGA EN LOS SISTEMAS MULTIUSUARIOS HAY UN FLUJO DE INTERRUPCIONES DIRECTAS DEL PROCESADOR. UNA RESPUESTA RAPIDA A ESTAS INTERRUPCIONES ES ESENCIAL PARA EL CUIDADO DEL BUEN USO DEL SISTEMA Y PROVEER TIEMPOS ACEPTABLES DE RESPUESTA DE LOS USUARIOS INTERACTIVOS.
LOS NUCLEOS DE INTERRUPTORES INHABILITADOS, MIENTRAS QUE ESTAN RESPONDIENDO A UNA INTERRUPCION; LAS INTERRUPCIONES SON INCAPACES DESPUES DE QUE EL PROCESO DE UNA INTERRUPCION HA SIDO COMPLETADO. CON UN FLUJO CONSTANTE DE INTERRUPCIONES, ES POSIBLE QUE EL NUCLEO PUEDA TENER INTERRUPCIONES INHABILITADAS POR UNA LARGA PORCION DE TIEMPO. POR LO TANTO EL PROCESAMIENTO ES POSIBLE EN CADA INTERRUPCION, Y DESPUES EL PROCESAMIENTO PERMANENTE DE CADA INTERRUPCION A UN APROPIADO SISTEMA DE PROCESO QUE PUEDE OPERAR MIENTRAS QUE EL NUCLEO SEA HABILITADO PARA INTERRUPCIONES ADICIONALES. ESTO ULTIMO SIGNIFICA QUE LAS INTERRUPCIONES PUEDEN SER HABILITADAS A UN PORCENTAJE MUCHO MAS LARGO DE TIEMPO, YA QUE EL SISTEMA TIENE MAS RESPUESTA.

3.12 SUMARIO DE LAS FUNCIONES DEL NUCLEO.

UN NUCLEO DEL SISTEMA OPERATIVO NORMALMENTE TIENE EL CODIGO PARA EJECUTAR LAS SIGUIENTES FUNCIONES:
- INTERRUPCION MANUAL
- CREACION Y DESTRUCCION DE PROCEOS
- ESTADO DE PROCESO DE SWITCHEO
- DESPACHAMIENTO
- SUSPENCION DE POCESOS Y RESUMEN
- COMUNICACION INTERPROCESO
- MANIPULACION DE PCB
- SOPORTE DE ACTIVIDADES I/O
- SOPORTE DE ALMACENAMIENTO ASIGNADO Y DESIGNADO
- SOPORTE DE ARCHIVO DE SISTEMA

- SOPORTE DE MECANISMO DE PROCEDURE CALL/RETURN
- SOPORTE DE CIERTAS FUNCIONES DE CONTEO

3.13 INTERRUPCIONES ASIGNADAS Y DESIGNADAS

EL NUCLEO ES NORMALMENTE ACCESADO POR UNA INTERRUPCION. EL NUCLEO INHABILITADO SE INTERRUMPE MIENTRAS QUE ESTE RESPONDE A LA INTERRUPCION. UNA VEZ QUE LA CAUSA DE LA INTERRUPCION HA SIDO DETERMINADA, EL NUCLEO PASA EL PROCESO DE LA INTERRUPCION A UN PROCESO DEL SISTEMA DESIGNADO QUE HA DE MANEJAR ESE TIPO DE INTERRUPCION.
EN ALGUNOS SISTEMAS, TODO EL PROCESO DE INTERRUPCION ES HECHO POR UN LARGO TIEMPO EN UNA PIEZA DEL S.O., EN ESOS SISTEMAS LAS INTERRUPCIONES SON HABILITADAS POR UN LARGO PORCENTAJE DE TIEMPO.

3.14 ESTRUCTURA JERARQUICA DEL SISTEMA

LA LITERATURA INCLUYE PAPELES QUE EVOCAN UN APROVECHAMIENTO JERARQUICO PARA DISENAR S.O. LA BASE DE LA JERARQUIA ES EL HARDWARE COMPUTACIONAL POR SI MISMO, ALGUNAS VECES LLAMADO "RAW MACHINE" O "NAKED IRON".
EN EL SIGUIENTE NIVEL SUPERIOR EN LA JERARQUIA SON VARIOS NUCLEO EN LOS SISTEMAS. ESTOS SON VISTOS COMO UNA CREACION DE UNA EXTENSION DE LA MAQUINA, UNA COMPUTADORA QUE NO SOLO OFRECE SU LENGUAJE MAQUINAL EN SOPORTE DEL S.O. Y SUS USUARIOS PARA UN GRUPO ADICIONAL DE CAPACIDADES PROVENIDA DE FUNCIONES DEL NUCLEO. ESTAS CAPACIDADES ADICIONALES SON LLAMADAS POR LO REGULAR PRIMITIVAS.ACERCA DEL NUCLEO EN LA JERARQUIA SON LOS PROCESOS DEL S.O. LOS QUE OPERAN EN EL SOPORTE DE LOS PROCESOS DE LOS USUARIOS POR EJEMPLO, EL SERVICIO DE SUPERVISION DE PROCESO AL QUE ACTUALMENTE SUPERVISA LAS OPERACIONES I/O PARA SERVICIO DE SISTEMAS A FAVOR DE VARIOS USUARIOS. EN EL TOPE DE LA JERARQUIA SON LOS USUARIOS LOS QUE SE PROCESAN A ELLOS MISMOS.
SEMEJANTES DISENOS JERARQUICOS SE HA APROVADO QUE SON FACILES DE DEBUG, MODIFICAR Y SE HAN APROVADO CORRECTAMENTE. EN DISENOS EN LOS CUALES EL NUCLEO ES ABIERTO POR SI MISMO SOBRE VARIOS NIVELES DE JERARQUIA, LA ELECCION DE CUAL FUNCION DEL NUCLEO SERA UBICADA EN EL NIVEL QUE LOS REQUIERA (PENSANDO CUIDADOSAMENTE). POR LO REGULAR EN SEMEJANTES DISENOS, LA RESTRICCION ES IMPUESTA TAL QUE SOLO LAS LLAMADAS EN DESCENDENTE SON ATENDIDAS; CADA NIVEL DEBE LLAMAR HACIA ARRIBA SOLO A ESAS FUNCIONES QUE SE VEAN EN EL NIVEL DIRECTAMENTE ABAJO.

3.15 MIGRACION DEL NUCLEO A MICROCODIGO.

UN CURSO DEFINIDO QUE HA EMERGIDO EN SISTEMAS RECIENTES ES

PARA EL ALOJAMIENTO DE MUCHOS DE LOS NUCLEOS EN MICROCODIGO. ESTA ES UNA EFECTIVA TECNICA DE SEGURIDAD, Y CON CUIDADO EL MICROCODIGO PUEDE HACER QUE LAS FUNCIONES DEL NUCLEO SE EFECTUEN MUCHO MAS RAPIDO.

PROCESOS CONCURRENTES ASINCRONICOS

4.1 INTRODUCCION

LOS PROCESOS SON CONCURRENTES SI EXISTEN A UN MISMO TIEMPO. LOS PROCESOS CONCURRENTES PUEDEN FUNCIONAR COMPLETAMENTE INDEPENDIENTES UNOS DE OTROS, O ELLOS PUEDEN SER ASINCRONOS, QUE SIGNIFICA QUE ELLOS REQUIEREN SINCRONIZACION Y COOPERACION OCASIONAL. LA ASINCRONIZACION ES UN TEMA COMPLEJO QUE MAS ADELANTE DISCUTIREMOS SU ORGANIZACION Y MANEJO DE SISTEMAS QUE MANTIENEN PROCESOS CONCURRENTES ASINCRONOS. MUCHOS MECANISMOS IMPORTANTES DE ASINCRONISMO SON PRESENTADOS. SUS SOLUCIONES SON PRESENTADAS COMO PROGRAMAS CONCURRENTES QUE SON NOTACIONES SIMILARES DE PROGRAMAS AUNQUE NO IDENTICOS AL DEL LENGUAJE CONCURRENTE PASCAL DESARROLLADO POR BRINCH HANSEN, OTRO LENGUAJE CONCURRENTE POPULAR BASADO EN PASCAL ES DESARROLLADO POR WRITH.

4.2 PROCESOS PARALELOS

A MEDIDA QUE EL HARDWARE DE COMPUTADORA CONTINUA DECRECIENDO EN TAMAÑO Y COSTO, HABRA UNA TENDENCIA CLARA HACIA EL MULTIPROCESO Y EVENTUALMENTE HACIA PARALELISMO MASIVO. SI CIERTAS OPERACIONES LOGICAS PUEDEN SER EJECUTADAS EN PARALELO, ENTONCES LAS COMPUTADORAS FUTURAS EJECUTARAN FISICAMENTE EN PARALELO, AUN SI EL NIVEL DE PARALELISMO SON MILES O TAL VEZ MILLONES DE ACTIVIDADES CONCURRENTES.
EL PROCESAMIENTO PARALELO ES IMPORTANTE POR VARIAS RAZONES. LA GENTE PREFIERE MEJOR ENFOCAR SU ATENCION HACIA UNA SOLA ACTIVIDAD A LA VEZ QUE PENSAR EN PARALELO. EL LECTOR PODRIA TRATAR DE LEER 2 LIBROS AL MISMO TIEMPO, LEYENDO UNA LINEA DE UNO, UNA LINEA DEL OTRO, LA SEGUNDA LINEA DEL SEGUNDO, DEL PRIMERO, ETC. ES DIFICIL EL DECIDIR CUALES ACTIVIDADES Y CUALES NO PUEDEN SER EJECUTADAS EN PARALELO. LOS PROGRAMAS EN PARALELO SON MUCHO MAS DIFICILES DE DEPURAR QUE LOS PROGRAMAS SECUENCIALES, DESPUES DE QUE UN ERROR ES SUPUESTAMENTE CORREGIDO, PUEDE SER IMPOSIBLE RECONSTRUIR LA SECUENCIA DE EVENTOS QUE MOSTRARON EL ERROR, TANTO QUE SERIA INAPROPIADA LA CERTIFICACION EN ALGUN SENTIDO QUE EL ERROR HA SIDO CORREGIDO.
LOS PROCESOS ASINCRONICOS DEBEN OCASIONALMENTE INTERACTUAR UNO CON OTRO Y ESTAS INTERACCIONES PUEDEN SER COMPLEJAS.
FINALMENTE, LOS PROGRAMAS PARALELOS SON MUCHO MAS DIFICILES DE PROBAR CORRECTAMENTE QUE LOS PROGRAMAS SECUENCIALES Y ES AMPLIAMENTE CREIDO QUE LA PRUEBA DE CORRECCION DEBE EVENTUALMENTE DESPLAZAR LA PRUEBA EXHAUSTIVA, SUS INTENTOS REALES VAN HA SER HECHOS EN DESARROLLAR SISTEMAS DE SOFTWARE

ALTAMENTE CONFIABLES.

4.3 ESTRUCTURAS DE CONTROL PARA INDICAR PARALELISMO: PARBEGIN/PAREND

MUCHAS CONSTRUCCIONES DE LENGUAJES DE PROGRAMACION PARA INDICAR PARALELISMO HAN APARECIDO EN LA LITERATURA. ESTOS GENERALMENTE INVOLUCRAN PARES DE ESTATUTOS COMO LOS SIGUIENTES:

- UN ESTATUTO QUE INDICA LA EJECUCION SECUENCIAL ES DIVIDIDA EN VARIAS SECUENCIAS DE EJECUCION PARALELA
- UN ESTATUTO QUE INDICA QUE CIERTAS SECUENCIAS DE EJECUCION PARALELA VAN A MEZCLARSE Y UNA EJECUCION SECUENCIAL VA A CONTINUAR.

ESTOS ESTATUTOS OCURREN NORMALMENTE EN PAREJAS Y SON COMUNMENTE LLAMADOS PARBEGIN/PAREND (PARA INICIO Y TERMINACION DE EJECUCIONES PARALELAS), O COBEGIN/COEND (PARA INICIO Y FIN DE EJECUCIONES CONCURRENTES). NOSOTROS USAMOS PARBEGIN/PAREND SUGERIDO POR DIJKSTRA. EN FORMA GENERAL VER LA SIGUIENTE FIGURA.

```
PARBEGIN
  ESTATUTO 1;
  ESTATUTO 2;
  ESTATUTO 3;
     .
     .
     .
  ESTATUTO N;
PAREND
```

FIGURA: CONSTRUCCIONES DE PARALELISMO PARBEGIN/PAREND.

SUPONGASE A UN PROGRAMA EJECUTANDO ACTUALMENTE UNA SOLA SECUENCIA DE INSTRUCCIONES Y ENCUENTRA LA CONSTRUCCION PARBEGIN, ESTO CAUSA QUE LA SECUENCIA DE EJECUCION SE DIVIDA EN VARIAS SECUENCIAS DE EJECUCION SEPARADOS, UNA PARA CADA ESTATUTO EN LA CONSTRUCCION PARBEGIN/PAREND. ESTO PUEDE SER CON UN ESTATUTO SIMPLE, LLAMADAS A PROCEDIMIENTOS, BLOCKS DE ESTATUTOS SECUENCIALES DELIMITADOS POR BEGIN/END, O LA COMBINACION DE ESTOS. CADA UNA DE LA SECUENCIA DE EJECUCION SEPARADA EVENTUALMENTE TERMINAN Y LLEGAN AL PAREND, CUANDO TODOS LAS SECUENCIAS PARALELAS DE CONTROL TERMINAN, UNA SOLA SECUENCIA DE EJECUCION ES RESUMED Y EL SISTEMA CONTINUA DESPUES DEL PAREND.

POR EJEMPLO, CONSIDEREMOS EL CALCULO DE UNA RAIZ CUADRADA EN UNA ECUACION CUADRATICA COMO LA SIGUIENTE:

```
X:=((-B+(B**2-4*A*C)**.5)/(2*A))
```

ESTA ASIGNACION PODRIA SER EVALUADA EN UN PROCESADOR SECUENCIAL (QUE POSEA UNA INSTRUCCION EXPONENCIAL) COMO SIGUE:

```
1 B**2
2 4*A
3 (4*A)*C
4 (B**2)-(4*A*C)
5 (B**2-4*A*C)**.5
6 -B
7 (-B)+((B**2-4*A*C)**.5)
8 2*A
9 ((-B+(B**2-4*A*C)**.5)/(2*A))
```

AQUI, CADA UNA DE LAS 9 OPERACIONES SON EJECUTADAS UNA POR UNA A LA VEZ, EN UNA SECUENCIA DETERMINADA POR LAS REGLAS DEL SISTEMA DE PROCEDENCIA DE OPERADORES.
EN UN SISTEMA QUE PODRIA SOPORTAR PROCESAMIENTO PARALELO, LA EXPRESION SERIA EVALUADA COMO SIGUE:

```
1   PARBEGIN
      TEMP1:=-B;
      TEMP2:=B**2;
      TEMP3:=4*A;
      TEMP4:=2*A;
    PAREND
2     TEMP5:=TEMP3*C;
3     TEMP5:=TEMP2-TEMP5;
4     TEMP5:=TEMP5**.5;
5     TEMP5:=TEMP1+TEMP5;
6     X:=TEMP5/TEMP4;
```

AQUI LAS 4 OPERACIONES DENTRO DE LA CONSTRUCCION PARBEGIN/PAREND SON EVALUADAS EN PARALELO, LAS RESTANTES 5 OPERACIONES DEBEN SER EJECUTADAS SECUENCIALMENTE. EJECUTANDO LOS CALCULOS EN PARALELO, ES POSIBLE REDUCIR EL TIEMPO REAL DE EJECUCION SUBSTANCIALMENTE.

4.4 EXCLUSION MUTUA

CONSIDERE UN SISTEMA CON MUCHAS TERMINALES DE TIEMPO COMPARTIDO. ASUMA QUE LOS USUARIOS TERMINAN CADA LINEA QUE ELLOS TECLEAN AL SISTEMA CON UN RETURN. SUPONGA QUE DESEA MONITOREAR CONTINUAMENTE EL NUMERO TOTAL DE LINEAS QUE LOS USUARIOS HAN INSERTADO DESDE QUE EMPEZO EL DIA, ASUMA QUE CADA TERMINAL DE USUARIO ES MONITOREADA POR UN DIFERENTE PROCESO. CADA VEZ QUE UNO DE ESTOS PROCESOS RECIBE UNA LINEA DESDE UNA TERMINAL DE USUARIO INCREMENTA UNA VARIABLE COMPARTIDA GLOBAL LINESENTERED (INSERTAR LINEAS), EN UNO.

CONSIDERE QUE PASA SI 2 PROCESOS INTENTAN INCREMENTAR SIMULTANEAMENTE LINESENTERED. ASUMA QUE CADA PROCESO TIENE SU PROPIA COPIA DE CODIGO.

```
LOAD LINESENTERED (INSERTAR LINEAS)
ADD 1
STORE LINESENTERED (INSERTAR LINEAS)
```

SUPONGA QUE LINESENTERED ES ACTUALMENTE 21687. AHORA SUPONGA QUE EL PRIMER PROCESO EJECUTA LAS INSTRUCCIONES LOAD Y ADD, DEJANDO ASI 21688 EN UN ACUMULADOR. LUEGO EL PROCESO CEDE EL PROCESADOR (POR MEDIO DE EXPIRACION DE QUANTUM) AL SEGUNDO PROCESO. EL SEGUNDO PROCESO EJECUTA AHORA LAS 3 INSTRUCCIONES PONIENDO ASI LINESENTERED EN 21688. ESTE CEDE EL PROCESADOR AL PRIMER PROCESO EL CUAL LUEGO CONTINUA EJECUTANDO LA INSTRUCCION STORE, TAMBIEN COLOCANDO 21688 EN LINESENTERED. DEBIDO AL ACCESO DESCONTROLADO DE LA VARIABLE COMPARTIDA LINESENTERED, EL SISTEMA HA PERDIDO EL RASTRO DE UNA DE LAS LINEAS YA QUE EL TOTAL CORRECTO DEBERIA DE HABER SIDO 21689.ESTE PROBLEMA PUEDE SER RESUELTO DANDO A CADA PROCESO UNA VARIABLE EXCLUSIVA LINESENTERED. MIENTRAS UN PROCESO INCREMENTA LA VARIABLE COMPARTIDA, LOS OTROS PROCESOS QUE DESEEN HACER ESTO EN EL MISMO MOMENTO DEBERAN ESPERAR; CUANDO ESE PROCESO HA TERMINADO EL ACCESO A LA VARIABLE COMPARTIDA, A UNO DE LOS PROCESOS ESPERANDO SE LE DEBE PERMITIR CONTINUAR. DE ESTE MODO CADA PROCESO ACCESANDO LOS DATOS COMPARTIDOS EXCLUYE TODOS LOS OTROS QUE SE EJECUTEN SIMULTANEAMENTE ESTO ES LLAMADO EXCLUSION MUTUA.

4.5 SECCIONES CRITICAS

LA EXCLUSION MUTUA DEBE SER IMPUESTA SOLO CUANDO LOS PROCESOS ACCESEN DATOS COMPARTIDOS, CUANDO LOS PROCESOS ESTEN EJECUTANDO OPERACIONES QUE NO ESTEN EN CONFLICTO DEBEN PERMITIRLES SEGUIR CONCURRENTEMENTE. CUANDO UN PROCESO ESTA ACCESANDO DATOS COMPARTIDOS, EL PROCESO SE DICE QUE ESTA EN SU SECCION CRITICA (O REGION CRITICA).
CLARAMENTE, PARA PREVENIR LA CLASE DE PROBLEMA DE LA ULTIMA SECCION, DEBE ASEGURARSE QUE CUANDO UN PROCESO ESTE EN SU SECCION CRITICA, TODOS LOS OTROS PROCESOS (AL MENOS AQUELLOS QUE ACCESEN LOS MISMOS DATOS COMPARTIDOS) SEAN EXCLUIDOS DE SUS PROPIAS SECCIONES CRITICAS.
MIENTRAS UN PROCESO ESTE EN SU SECCION CRITICA, OTROS PROCESOS PUEDEN CIERTAMENTE CONTINUAR EJECUTANDOSE FUERA DE SUS SECCIONES CRITICAS. CUANDO UN PROCESO DEJA SU SECCION CRITICA, LUEGO A OTRO PROCESO QUE ESPERA ENTRAR EN SU PROPIA SECCION CRITICA LE DEBE SER PERMITIDO SEGUIR (SI VERDADERAMENTE HAY UN PROCESO ESPERANDO). LA IMPOSICION DE LA EXCLUSION MUTUA ES UNO DE LOS PROBLEMAS CLAVES EN LA

PROGRAMACION CONCURRENTE. MUCHAS SOLUCIONES HAN SIDO IDEADAS: ALGUNAS SOLUCIONES DE SOFTWARE Y ALGUNAS SOLUCIONES DE HARDWARE; ALGUNAS MAS BIEN DE BAJO NIVEL Y ALGUNAS DE ALTO NIVEL; ALGUNOS REQUIEREN COOPERACION VOLUNTARIA ENTRE PROCESOS Y ALGUNOS DEMANDAN ADHERENCIA RIGIDA A PROTOCOLOS ESTRICTOS. ESTAR DENTRO DE UNA SECCION CRITICA ES UN MUY ESPECIAL AJUSTADO AL PROCESO. EL PROCESO TIENE ACCESO EXCLUSIVO A DATOS COMPARTIDOS Y TODOS LOS OTROS PROCESOS QUE ACTUALMENTE REQUIEREN ACCESO A ESOS DATOS SE MANTIENEN ESPERANDO. POR LO TANTO LAS SECCIONES CRITICAS DEBEN EJECUTAR TAN RAPIDAMENTE COMO SEA POSIBLE, UN PROCESO NO DEBE OBSTRUIR DENTRO DE SU SECCION CRITICA Y LAS SECCIONES CRITICAS DEBEN SER CUIDADOSAMENTE CODIFICADAS (PARA EVITAR LA POSIBILIDAD DE CICLOS INFINITOS POR EJEMPLO).
SI UN PROCESO EN UNA SECCION CRITICA TERMINA YA SEA VOLUNTARIA O INVOLUNTARIAMENTE, ENTONCES EL S.O. AL EFECTUAR SU CONTABILIDAD DE TERMINACION DEBE LIBERAR LA EXCLUSION MUTUA PARA QUE OTROS PROCESOS PUEDAN ENTRAR A SUS SECCIONES CRITICAS.

4.6 PRIMITIVOS DE EXCLUSION MUTUA

EL PROGRAMA CONCURRENTE DE LA SIGUIENTE FIGURA, IMPLEMENTO CORRECTAMENTE EL MECANISMO DE CONTEO DE LINEAS DE LA SECCION PREVIA. POR SIMPLICIDAD, ASUMIMOS SOLO 2 PROCESOS CONCURRENTES EN LOS PROGRAMAS PRESENTADOS EN ESTE Y EN VARIAS SECCIONES SIGUIENTES MANEJAR "N" PROCESOS CONCURRENTES ES CONSIDERABLEMENTE MAS COMPLEJO.
LAS CONSTRUCCIONES ENTRADAXCLUSIONMUTUA (ENTERMUTUALEXCLUSION) Y SALIDA-EXCLUSION-MUTUA (EXITMUTUALEXCLUSION) INTRODUCIDAS EN LA SIGUIENTE FIGURA, ENCAPSULAN EL CODIGO DE CADA PROCESO QUE ACCESA LA VARIABLE COMPARTIDA LINESENTERED, ESTO ES, ESTAS CONSTRUCCIONES DELINEAN LAS SECCIONES CRITICAS. ESTAS OPERACIONES SON ALGUNAS VECES LLAMADAS PRIMITIVOS DE EXCLUSION MUTUA, ESTO ES, ELLOS ENVUELVEN TODAS LAS OPERACIONES MAS FUNDAMENTALES INHERENTES A LA EXCLUSION MUTUA.

```
PROGRAM MUTUALEXCLUSION;
  VAR
   LINESENTERED:INTEGER
  PROCEDURE PROCESSONE;
   WHILE TRUE  DO
   BEGIN
    GETNEXTLINEFROMTERMINAL;
    ENTERMUTUALEXCLUSION;
    LINESENTERED:= LINESENTERED + 1;
    EXITMUTUALEXCLUSION;
    PROCESSTHEOLINE;
```

```
 END;
PROCEDURE PROCESSTWO;
 WHILE TRUE DO
 BEGIN
  GETNEXTLINEFROMTERMINAL;
  ENTERMUTUALEXCLUSION;
  LINESENTERED:= LINESENTERED + 1;
  EXITMUTUALEXCLUSION;
  PROCESSTHELINE;
 END;
BEGIN
 LINESENTERED:= 0;
 PARBEGIN
  PROCESSONE;
  PROCESSTWO;
 PAREND
END.
```

FIGURA: USO PRIMITIVO DE EXCLUSION MUTUA.

EN EL CASO DE 2 PROCESOS, ESTOS PRIMITIVOS OPERAN COMO SIGUE.
CUANDO EL PROCESO UNO (PROCESSONE) EJECUTA ENTERMUTUALEXCLUSION, SI EL PROCESO DOS (PROCESSTWO) NO ESTA EN SU SECCION CRITICA, ENTONCES PROCESSONE ENTRA A SU SECCION CRITICA, ACCESANDO LA VARIABLE COMPARTIDA Y LUEGO EJECUTA EXITMUTUALEXCLUSION PARA INDICAR QUE ESTE HA DEJADO SU SECCION CRITICA.
SI PROCESSTWO ESTA EN SU SECCION CRITICA CUANDO PROCESSONE EJECUTA ENTERMUTUALEXCLUSION, ENTONCES PROCESSONE ESPERA HASTA QUE PROCESSTWO EJECUTA EXITMUTUALEXCLUSION. PROCESSONE PUEDE ENTONCES SEGUIR Y ENTRAR A SU SECCION CRITICA.
SI PROCESSONE Y PROCESSTWO SIMULTANEAMENTE EJECUTAN ENTERMUTUALEXCLUSION, ENTONCES A UNO SE LE FERMITIRA SEGUIR Y UNO PERMANECERA ESPERANDO, ASUMIREMOS QUE EL GANADOR ES SELECCIONADO AL AZAR.

4.7 IMPLEMENTACION DE EXCLUSIONES MUTUAS PRIMITIVAS

LLAMAMOS A UNA IMPLEMENTACION DE ENTRADA-EXCLUSION-MUTUA (ENTERMUTUALEXCLUSION) Y SALIDA-EXLUSION-MUTUA (EXITMUTUALEXCLUSION) A LA QUE SATISFACE LAS SIGUIENTES 4 RESTRICCIONES:
- LA SOLUCION ES IMPLEMENTADA PURAMENTE EN SOFTWARE EN UNA MAQUINA SIN INSTRUCCIONES DE EXCLUSION MUTUA ESPECIFICAMENTE DISENADAS.
CADA INSTRUCCION DEL LENGUAJE MAQUINAL ES EJECUTADA INDIVISIBLEMENTE, ESTO ES, CADA VEZ QUE SE EMPIEZA UNA INSTRUCCION SE TERMINA SIN INTERRUPCION. SI VARIOS PROCESADORES TRATAN DE ACCESAR AL MISMO TIEMPO DATOS,

ASUMIREMOS QUE UNA CARACTERISTICA DEL HARDWARE LLAMADA ALMACENAMIENTO INTERRUMPIDO ('STORAGE INTERLOCK') RESOLVERA CUALQUIER PROBLEMA.

EL STORAGE INTERLOCK SECUENCIALIZA LAS CONSULTAS CONFLICTIVAS DE LOS DISTINTOS PROCESADORES, ES DECIR SUCEDE UNA CONSULTA A LA VEZ.

- NO SE HARAN SUPOSICIONES ACERCA DE LAS VELOCIDADES RELATIVAS DE LOS PROCESOS CONCURRENTES ASINCRONOS.
- LOS PROCESOS OPERANDO FUERA DE SUS SECCIONES CRITICAS NO PUEDEN PREVENIR A OTROS PROCESOS DE ENTRAR A SUS SECCIONES CRITICAS.
- A LOS PROCESOS NO SE LES DEBE POSPONER INDEFINIDAMENTE EL ENTRAR A SU SECCION CRITICA.

UNA INPLEMENTACION DE SOFTWARE ELEGANTE DE EXCLUSIONES MUTUAS FUE PRESENTADA PRIMERO POR EL DUQUE MATEMATICO DEKKER.

EN LA SIGUIENTE SECCION SEGUIMOS EL DESARROLLO DE DIJKSTRA DEL ALGORITMO DE DEKKER.

4.8 ALGORITMO DE DEKKER

LA SIGUIENTE FIGURA MUESTRA EL PRIMER ESFUERZO AL ESPECIFICAR EL CODIGO PARA LA EJECUCION DE EXCLUSION MUTUA EN EL CONTEXTO DE UN PROGRAMA CONCURRENTE CON 2 PROCESOS. LAS CONSTRUCCIONES PARBEGIN/PAREND OCASIONAN QUE PROCESSONE Y PROCESSTWO OPEREN COMO PROCESOS CONCURRENTES. CADA UNO DE ESTOS PROCESOS SE CICLA INDEFINIDAMENTE, ENTRANDO Y VOLVIENDO A ENTRAR A SU SECCION CRITICA REPETIDAMENTE. EN LA SIGUIENTE FIGURA ES IMPLEMENTADA LA ENTERMUTUALEXCLUSION CON UN SOLO CICLO WHILE QUE SE MANTIENE CICLADO HASTA QUE EL NUMERO DE PROCESSNUMBER ES IGUAL AL NUMERO DEL PROCESO, LA EXITMUTUALEXCLUSION ES IMPLEMENTADA COMO UNA SOLA INSTRUCCION EN LA CUAL A PROCESSNUMBER SE LE DA EL VALOR DEL NUMERO DEL OTRO PROESO.

PROCESSONE EJECUTA EL WHILE DO, COMO PROCESSNUMBER INICIALMENTE ES UNO, ENTONCES PROCESSONE ENTRA A SU SECCION CRITICA. PROCESSTWO ENCUENTRA QUE PROCESSNUMBER ES IGUAL A 1 Y SE MANTIENE ENCENDIDO EN SU WHILE DO. CADA VEZ QUE PROCESSTWO TOMA AL PROCESADOR, SIMPLEMENTE SE CICLA ESPERANDO QUE PROCESSNUMBER TOME EL VALOR DE 2, ASI PROCESSTWO NO ENTRA A SU SECCION CRITICA Y LA EXCLUSION MUTUA QUEDA GARANTIZADA.

EVENTUALMENTE, PROCESSONE TERMINA LA EJECUCION DE SU SECCION CRITICA (NO DEBEMOS TOMAR A LOS LOOPS COMO INFINITOS) Y COLOCA EN PROCESSNUMBER UN 2 PARA PERMITIR LA ENTRADA A PROCESSTWO A SU SECCION CRITICA. PROCESSONE DEBE IR PRIMERO, ASI ES QUE SI PROCESSTWO ESTA LISTO PARA ENTRAR A SU SECCION CRITICA, DEBE SER RETARDADO CONSIDERABLEMENTE.

DESPUES DE QUE PROCESSONE ENTRA Y SALE DE SU SECCION CRITICA, DEBE SEGUIR PROCESSTWO, AUNQUE PROCESSONE QUEDRA VOLVER A ENTRAR Y PROCESSTWO NO ESTE LISTO AUN. DE AQUI QUE LOS PROCESOS DEBEN ENTRAR Y SALIR DE SU SECCION CRITICA ALTERNANDOSE EN UNA FORMA ESTRICTA.
SI UN PROCESO SE NECESITA HACER MUCHAS VECES MAS FRECUENTEMENTE QUE EL OTRO, SE VE RESTRINGIDO A OPERAR A UNA VELOCIDAD MUCHO MAS LENTA DE LA QUE REQUIERE. EL SISTEMA NO PUEDE CONVERTIRSE COMPLETAMENTE EN ESTANCADOR, AL MENOS UN PROCESO PUEDE PROCEDER SI AMBOS ESTAN INTENTANDO SIMULTANEAMENTE ENTRAR A SU SECCION CRITICA. SI UNO DE LOS PROCESOS ES TERMINADO, ENTONCES EVENTUALMENTE AL OTRO PROCESO NO LE SERA PERMITIDO PROSEGUIR.

```
PROGRAM VERSIONONE;
VAR PROCESSNUMBER:INTEGER;
PROCEDURE PROCESSONE;
BEGIN
 WHILE TRUE DO
 BEGIN
  WHILE PROCESSNUMBER=2 DO
  CRITICALSECTIONONE;P
  PROCESSNUMBER:=2;
  OTHERSTUFFONE
 END
END;
PROCEDURE PROCESSTWO;
BEGIN
 WHILE TRUE DO
 BEGIN
  WHILE PROCESSNUMBER=1 DO
  CRITICALSECTIONTWO;
  PROCESSNUMBER:=1;
  OTHERSTUFFTWO;
 END
END;
BEGIN
 PROCESSNUMBER:=1;
 PARBEGIN
  PROCESSONE;
  PROCESSTWO
 PAREND
END.
```

FIGURA: VERSION 1 DE PRIMITIVAS DE EXCLUSION MUTUA.

EN LA PRIMERA SOLUCION HABIA SOLAMENTE UNA VARIABLE GLOBAL Y ESTA FORZABA A UN ENCERRAMIENTO DE LA SINCRONIZACION COMO UN PROBLEMA. POR ESO EN LA VERSION 2 (SIGUIENTE FIGURA) USAMOS

2 VARIABLES, P1INSIDE QUE ES VERDADERA SI PROCESSONE ESTA DENTRO DE SU SECCION CRITICA, Y P2INSIDE QUE ES VERDADERO SI PROCESSTWO ESTA DENTRO DE SU SECCION CRITICA.
AHORA PROCESSONE SE MANTIENE ESTANCADO EN UNA ESPERA MIENTRAS QUE P2INSIDE SEA VERDADERO. EVENTUALMENTE PROCESSTWO DEJA SU SECCION CRITICA Y EJECUTA SU PROPIO CODIGO DE SALIDA DE EXCLUSION MUTUA COLOCANDO A P2INSIDE FALSO. ENTONCES PROCESSONE COLOCA VERDADERO A P1INSIDE Y ENTRA A SU SECCION CRITICA. MIENTRAS QUE P1INSIDE PERMANEZCA VERDADERO, PROCESSTWO NO PUEDE ENTRAR A SU SECCION CRITICA.
UNA VEZ MAS, SALE A LA SUPERFICIE LO INSIDIOSO DE LA PROGRAMACION CONCURRENTE. COMO PROCESSONE Y PROCESSTWO SON PROCESOS CONCURRENTES, AMBOS PODRIAN ATENDER A SUS RESPECTIVAS SECUENCIAS DE CODIGO DE ENTRADA DE EXCLUSION MUTUA SIMULTANEAMENTE. AL INICIO, AMBAS P1INSIDE Y P2INSIDE SON FALSAS. PROCESSONE PODRIA PROBAR P2INSIDE Y ENCONTRARLO FALSO, ENTONCES ANTES DE QUE PROCESSONE PUDIERA COLOCAR VERDADERO A P1INSIDE, PROCESSTWO PODRIA PROBAR P1INSIDE Y ENCONTRARLO FALSO. EN ESTE PUNTO PROCESSTWO COLOCA P1INSIDE VERDADERO Y ENTRA A SER SECCION CRITICA, Y PROCESSTWO COLOCA VERDADERO A P2INSIDE Y ENTRA A SU SECCION CRITICA. AMBOS PROCESOS ESTAN EN SUS SECCIONES CRITICAS SIMULTANEAMENTE, ASI ES QUE LA VERSION 2 NO GARANTIZA LA EXCLUSION MUTUA.

```
PROGRAM VERSIONTWO;
  VAR P1INSIDE,P2INSIDE:BOOLEAN;
  PROCEDURE PROCESSONE;
  BEGIN
   WHILE TRUE DO
   BEGIN
    WHILE P2INSIDE DO
    P1INSIDE:=TRUE;
    CRITICALSECTIONONE;
    P1INSIDE:=FALSE;
    OTHERSTUFFONE
   END
  END;
  PROCEDURE PROCESSTWO;
  BEGIN
   WHILE TRUE DO
   BEGIN
    WHILE P1INSIDE DO
    P2INSIDE:=TRU;
    CRITICALSECTIONTWO;
    P2INSIDE:=FALSE;
    OTHERSTUFFTWO
   END
  END;
  BEGIN
```

```
   P1INSIDE:=FALSE;
   P2INSIDE:=FALSE;
   PARBEGIN
    PROCESSONE;
    PROCESSTWO
   PAREND
  END.
```

FIGURA: VERSION 2 DE PRIMITIVOS DE EXCLUSION MUTUA.

EN LA VERSION 2 HABIA DIFICULTAD PORQUE ENTRE EL TIEMPO QUE UN PROCESO DETERMINABA EN EL WHILE TEST SI PODRIA SEGUIR ADELANTE Y EL TIEMPO QUE EL PROCESO COLOCABA UNA BANDERA PARA DECIR QUE ESTABA EN SU SECCION CRITICA, HAY UN TIEMPO SUFICIENTE PARA QUE EL OTRO PROCESO INSPECCIONE SU BANDERA Y SE DESLICE A SU SECCION CRITICA. DE AQUI, UNA VEZ QUE UN PROCESO ATENTA AL WHILE TEST SE DEBE ASEGURAR QUE EL OTRO PROCESO NO PROCEDA A PASAR A TRAVES DE SU PROPIO WHILE TEST. LA VERSION 3 (SIGUIENTE FIGURA) INTENTA RESOLVER ESTO COLOCANDO UNA BANDERA EN CADA PROCESO ANTES DE EJECUTAR EL WHILE. UN PROBLEMA HA SIDO RESUELTO, PERO HA SEGUIDO OTRO (QUE BIEN) CADA PROCESO ENCIENDE SU BANDERA ANTES DE PASAR AL WHILE TEST, CADA PROCESO ENCONTRARA ENCENDIDA LA BANDERA DEL OTRO Y SE QUEDA PARA SIEMPRE EN EL WHILE DO. ESTE ES UN EJEMPLO DE ESTANCAMIENTO DE 2 PROCESOS.
EL PROBLEMA CON LA VERSION 3 ES QUE CADA UNO DE LOS PROCESOS PUEDE ESTANCARSE EN SU RESPECTIVO CICLO DE WHILE DO. NECESITAMOS UNA FORMA DE ROMPER CON ESTOS CICLOS.

```
PROGRAM VERSIONTHREE;
  VAR P1WONTSTOENTER,P2WONTSTOENTER:BOOLEAN;
  PROCEDURE PROCESSONE;
  BEGIN
   WHILE TRUE DO
   BEGIN
    P1WONTSTOENTER:=TRUE;
    WHILE P2WANTSTOENTER DO;
    CRITICALSECTIONONE;
    P1WANTSTOENTER:=FALSE;
    OTHERSTUFFONE
   END
  END;
  PROCEDURE PROCESSTWO;
  BEGIN
   WHILE TRUE DO
   BEGIN
    P2WANTSTOENTER:=TRUE;
    WHILE P1WANTSTOENTER DO
    CRITICALSECTIONTWO;
```

```
      P2WANTSTOENTER:=FALSE;
      OTHERSTUFFTWO
     END
   END;
   BEGIN
    P1WANTSTOENTER:=FALSE;
    P2WANTSTOENTER:=FALSE;
    PARBEGIN
     PROCESSONE;
     PROCESSTWO
    PAREND
   END
END.
```

FIGURA: VERSION 3 DE PRIMITIVOS DE EXCLUSION MUTUA.

LA VERSION 4 (SIGUIENTE FIGURA) REALIZA ESTO FORZANDO A CADA PROCESO DE CICLADO A PONER SU BANDERA EN FALSO REPETIDAMENTE POR BREVES PERIODOS, ESTO LE PERMITIRA AL OTRO PROCESO A PASAR A SU WHILE CON SU PROPIA BANDERA ENCENDIDA.
LA EXCLUSION MUTUA ESTA GARANTIZADA Y NO PUEDE OCURRIR UN ESTANCAMIENTO, PERO OTRO PROBLEMA POTENCIALMENTE DEVASTADOR PODRIA DESARROLLARSE LLAMADO POSPOSICION INDEFINIDA (LET'S SEE HOW) COMO NO PODEMOS HACER NINGUNA CONCLUSION ACERCA DE LA VELOCIDAD RELATIVA DE LOS PROCESOS CONCURRENTES ASINCRONOS, DEBEMOS CONSIDERAR TODAS LAS SECUENCIAS DE EJECUCION POSIBLES. CADA PROCESO PUEDE PONER SU BANDERA EN VERDADERA, DESPUES HACE EL TEST DEL WHILE, DESPUES ENTRA AL LOOP WHILE, DESPUES PONE SU BANDERA EN FALSO, LUEGO PONE SU BANDERA EN VERDADERO Y DESPUES REPITE LA SECUENCIA COMENZANDO CON EL TEST DEL WHILE. MIENTRAS HACEN ESTO, LAS CONDICIONES DE PROBADO QUEDARAN VERDADERAS. POR SUPUESTO, DICHA OPERACION TIENE MUY POCA PROBABILIDAD DE OCURRIR PERO PODRIA OCURRIR. DE AQUI QUE LA VERSION 4 QUEDA RECHAZADA.
SI UN SISTEMA ESTA USANDO ESTE TIPO DE EXCLUSION MUTUA ESTARIA CONTROLANDO UN VUELO ESPACIAL, UN MARCAPASO DE CORAZON, O UN SISTEMA DE CONTROL DE TRAFICO AEREO, LA POSIBILIDAD DE UNA POSPOSICION INDEFINIDA Y LA CONSECUENTE FALLA DEL SISTEMA SERIA INEVITABLE.

```
PROGRAM VERSIONFOUR;
 VAR P1WANTSTOENTER,P2WANTSTOENTER:BOOLEAN;
 PROCEDURE PROCESSONE;
 BEGIN
  WHILE TRUE DO
  BEGIN
   P1WANTSTOENTER:=TRUE;
   WHILE P2WANTSTOENTER DO
   BEGIN
```

```
    P1WANTSTOENTER:=FALSE;
    DELAY(RANDOM,FEWCYCLES);
    P1WANTSTOENTER:=TRUE
   END;
   CRITICALSECTIONONE;
   P1WANTSTOENTER:= FALSE;
   OTHERSTUFFONE
  END
END;
PROCEDURE PROCESSTWO;
BEGIN
 WHILE TRUE DO
 BEGIN
  P2WANTSTOENTER:=TRUE;
  WHILE P1WANTSTOENTER DO
  BEGIN
   P2WANTSTOENTER:=FALSE;
   DELAY(RANDOM,CYCLES);
   P2WANTSTOENTER:=TRUE
  END
  CRITICALSECTIONTWO;
  P2WANTSTOENTER:=FALSE;
  OTHERSTUFFTWO
 END
END;
BEGIN
 P1WANTSTOENTER:=FALSE;
 P2WANTSTOENTER:=FALSE;
 PARBEGIN
  PROCESSONE;
  PROCESSTWO
 PAREND
END.
```

FIGURA: VERSION 4 DE PRIMITIVOS DE EXCLUSION MUTUA.

EN SOLO UNAS CUANTAS LINEAS DE CODIFICACION, EL ALGORITMO DE DEKKER (SIGUIENTE FIGURA) MANEJA ELEGANTEMENTE LA EXCLUSION MUTUA DE 2 PROCESOS SIN LA NECESIDAD DE INSTRUCCIONES ESPECIALES EN HARDWARE.
EL ALGORITMO DE DEKKER RESUELVE LA POSIBILIDAD DE POSPOSICION INDEFINIDA EXPERIMENTADA EN LA VERSION 4. PROCESSONE INDICA SU DESEO DE ENTRAR A SU SECCION CRITICA ENCENDIENDO SU BANDERA, ENTONCES PROCEDE AL TEST DEL WHILE DONDE CHECA SI PROCESSTWO TAMBIEN QUIERE ENTRAR, SI LA BANDERA DE PROCESSTWO ESTA APAGADA, ENTONCES PROCESSONE SALTA EL PARRAFO DEL WHILE LOOP Y ENTRA A SU SECCION CRITICA, SUPONGA DE TODAS MANERAS QUE CUANDO PROCESSONE EJECUTA EL TEST DEL WHILE, DESCUBRE QUE LA BANDERA DE

PROCESSTWO ESTA ENCENDIDA. ESTO HACE QUE PROCESSONE ENTRE AL PARRAFO DE SU WHILE LOOP. AQUI SE FIJA EN LA VARIABLE FAVOREDPROCESS, QUE ES USADA PARA RESOLVER LOS CONFLICTOS QUE SURGEN CUANDO AMBOS PROCESOS QUIEREN ENTRAR A SU SECCION CRITICA SIMULTANEAMENTE. SI PROCESSONE ES EL PROCESO FAVORECIDO, SALTA EL PARRAFO DEL IF Y EJECUTA RAPIDAMENTE EL TEST DEL WHILE ESPERANDO QUE PROCESSTWO APAGUE SU BANDERA (ADELANTE VEREMOS QUE PROCESSTWO HARA ESTO EVENTUALMENTE). SI PROCESSONE DETERMINA QUE PROCESSTWO ES EL PROCESO FAVORECIDO, ENTONCES PROCESSONE ES FORZADO A ENTRAR AL PARRAFO DEL IF DONDE APAGA SU PROPIA BANDERA, DESPUES SE CICLA DENTRO DEL WHILE DO MIENTRAS QUE PROCESSTWO PERMANEZCA SIENDO EL PROCESO FAVORECIDO. APAGANDO SU BANDERA PROCESSONE PERMITE QUE PROCEETWO ENTRE A SU SECCION CRITICA. EVENTUALMENTE PROCESSTWO ABANDONA SU SECCION CRITICA Y EJECUTA SU CODIGO DE SALIDA DE EXCLUSION MUTUA. ESTOS ESTATUTOS COLOCAN A PROCESSONE COMO EL PROCESO FAVORECIDO Y APAGA LA BANDERA DE PROCESSTWO. AHORA PROCESSONE PASA O SALE DE SU WHILE INTERNO Y ENCIENDE SU PROPIA BANDERA.
ENTONCES PROCESSONE EJECUTA EL TEST DEL WHILE EXTERNO. SI LA BANDERA DE PROCESSTWO (LA CUAL FUE RECIENTEMENTE APAGADA) SIGUE APAGADA, ENTONCES PROCESSONE ENTRA A SU SECCION CRITICA. SI DE TODOS MODOS PROCESSTWO TRATO RAPIDAMENTE DE ENTRAR DE NUEVO A SU SECCION CRITICA, ENTONCES LA BANDERA DE PROCESSTWO ESTARA ENCENDIDA Y PROCESSONE SE VERA UNA VEZ MAS FORZADA A ENTRAR AL WHILE EXTERNO. PERO ESTA VEZ PROCESSONE SE ENCUENTRA EN "EL LUGAR DEL CONDUCTOR" PORQUE AHORA ES EL PROCESO FAVORECIDO (RECUERDA QUE CUANDO PROCESSTWO DEJA SU SECCION CRITICA, COLOCA FIRST EN FAVORED PROCESS). ASI PROCESSONE SALTA AL PARRAFO DEL IF, Y EJECUTA REPETIDAMENTE EL TEST DEL WHILE EXTERNO HASTA QUE PROCESSTWO "HUMILDEMENTE" APAGA SU BANDERA PERMITIENDO ASI QUE PROCESSONE ENTRE A SU SECCION CRITICA.
CONSIDERE LA SIGUIENTE POSIBILIDAD INTERESANTE, A MEDIDA QUE PROCESSONE SALE DE SU LOOP DE ESPERA INTERNO, ES POSIBLE PARA ESTE PERDER EL PROCESADOR Y TAMBIEN ES POSIBLE QUE PROCESSTWO, DESPUES DE VARIOS CICLOS, QUIERA INTENTAR DE NUEVO ENTRAR A SU SECCION CRITICA. ENTONCES PROCESSTWO ENCENDERA SU BANDERA PRIMERO Y VOLVERA A ENTRAR A SU SECCION CRITICA. CUANDO PROCESSONE TIENE DE NUEVO AL PROCESADOR, ENCIENDE SU BANDERA YA QUE AHORA SERA EL TURNO PARA PROCESSONE, SI PROCESSTWO TRATA DE VOLVER A ENTRAR, TENDRIA QUE APAGAR SU BANDERA Y SE VERA FORZADO A ENTRAR A SU LOOP DE ESPERA INTERNO Y ASI PROCESSONE PODRA ENTRAR A SU SECCION CRITICA. ASI ESTA TRIQUINUELA YA NO RESULTARA EN UNA POSPOSICION INDEFINIDA.

```
PROGRAM ALGORITMODEDEKKER;
VAR FAVOREDPROCESS: (FIRST,SECOND);
```

```
P1WANTSTOENTER,P2WANTSTOENTER:BOOLEAN;
PROCEDURE PROCESSONE;
BEGIN
 WHILE TRUE DO
 BEGIN
  P1WANTSTOENTER:=TRUE;
  WHILE P2WANTSTOENTER DO
  IF FAVOREDPROCESS = SECOND THEN
  BEGIN
   P1WANTSTOENTER:=FALSE
   WHILE FAVOREDPROCESS = SECOND DO
   P1WANTSTOENTER:= TRUE
  END;
  CRITICALSECTIONONE;
  FAVOREDPROCESS:= SECOND;
  P1WANTSTOENTER:= FALSE;
  OTHERSTUFFONE
 END
END;
PROCEDURE PROCESSTWO;
BEGIN
 WHILE TRUE DO
 BEGIN
  P2WANTSTOENTER:=TRUE;
  WHILE P1WANTSTOENTER DO
  IF FAVOREDPROCESS = FIRST THEN
  BEGIN
   P2WANTSTOENTER:= FALSE;
   WHILE FAVOREDPROCESS = FIRST DO
   P2WANTSTOENTER:= TRUE
  END;
  CRITICALSECTIONTWO;
  FAVOREDPROCESS:= FIRST;
  P2WANTSTOENTER:=FALSE;
  OTHERSTUFFTWO
 END
END;
BEGIN
 P1WANTSTOENTER:= FALSE;
 P2WANTSTOENTER:= FALSE;
 FAVOREDPROCESS:=FIRST;
 PARBEGIN
  PROCESSONE;
  PROCESSTWO
 PAREND
END.
```

FIGURA: ALGORITMO DE DEKKER PARA LA IMPLEMENTACION DE PRIMITIVOS DE EXCLUSION MUTUA.

4.9 EXCLUSION MUTUA PARA N-PROCESOS

DIJKSTRA FUE EL PRIMERO EN PRESENTAR UNA SOLUCION EN SOFTWARE PARA LA IMPLEMENTACION DE PRIMITIVOS DE EXCLUSION MUTUA PARA N PROCESOS. KNUTH RESPONDIO CON UNA SOLUCION QUE ELIMINABA LA POSIBILIDAD DE LA POSPOSICION INDEFINIDA EN EL ALGORITMO DE DIJKSTRA, PERO UN PROCESO SEGUIA EXPERIMENTANDO UN RETRASO MUY LARGO. ESTO GENERO EN UNA SERIE DE ESFUERZOS PARA ENCONTRAR ALGORITMOS CON RETRASOS MAS CORTOS. EISENBERG Y McGUIRE PRESENTARON UNA SOLUCION GARANTIZANDO QUE UN PROCESO ENTRARIA A SU SECCION CRITICA EN N-1 INTENTOS. LAMPART DESARROLLO UNA SOLUCION QUE ES PARTICULARMENTE APLICADA A SISTEMAS DE PROCESOS DISTRIBUIDOS. EL ALGORITMO USA UN SISTEMA DE "TOMA UN BOLETO" COMO EL QUE SE EMPLEA EN LAS PASTELERIAS MUY CONCURRIDAS Y SE LE HA APODADO ALGORITMO DE PASTELERIA DE LAMPART (LAMPART'S BAKEY ALGORITHM). BRINCH HANSEN TAMBIEN DISCUTIA EL CONTROL DE LA CONCURRENCIA ENTRE PROCESOS DISTRIBUIDOS.

4.10 UNA SOLUCION EN HARDWARE PARA LA EXCLUSION MUTUA: LA INSTRUCCION TESTANDSET

EL ALGORITMO DE DEKKER ES UNA SOLUCION EN SOFTWARE PARA EL PROBLEMA DE LA EXCLUSION MUTUA. ESTA SECCION PRESENTA UNA SOLUCION EN HARDWARE.
LA CLAVE PARA TENER EXITO AQUI ES TENER UNA SOLA INSTRUCCION EN HARDWARE QUE LEEA UNA VARIABLE, ALMACENE SU VALOR EN UNA AREA SEGURA Y COLOQUE UN CIERTO VALOR A ESA VARIABLE. ESTA INSTRUCCION, A MENUDO LLAMADA TESTANDSET, UNA VEZ INICIADA COMPLETARA TODAS ESTAS FUNCIONES SIN INTERRUPCION. LA INSTRUCCION INDIVISIBLE TESTANDSET

```
TESTANDSET (A,B)
```

LEE EL VALOR BOOLEANO DE B, LO COPIA EN A Y DESPUES COLOCA VERDADERO (TRUE) EN B TODO DENTRO DE UNA SOLA INSTRUCCION INTERRUMPIDA. TESTANDSET PUEDE SER USADA COMO EN LA SIUIENTE FIGURA PARA REALIZAR LA EXCLUSION MUTUA:

```
PROGRAM TESTANDSETEXAMPLE;
 VAR ACTIVE:BOOLEAN;
 PROCEDURE PROCESSONE;
 VAR ONECANNOTENTER:BOOLEAN;
 BEGIN
  WHILE TRUE DO
  BEGIN
   ONECANNOTENTER:=TRUE;
   WHILE ONECANNOTENTER DO
   TESTANDSET(ONECANNOTENTER,ACTIVE);
```

```
    CRITICALSECTIONONE;
    ACTIVE:=FALSE;
    OTHERSTUFFONE
   END
  END;
  PROCEDURE PROCESSTWO;
  VAR TWOCANNOTENTER:BOOLEAN;
  BEGIN
   WHILE TRUE DO
   BEGIN
    TWOCANNOTENTER:= TRUE;
    WHILE TWOCANNOTENTER DO
     TESTANDSET(TWOCANNOTENTER,ACTIVE);
    CRITICALSECTIONTWO;
    ACTIVE:=FALSE;
    OTHERSTUFFTWO
   END
  END;
  BEGIN
   ACTIVE:=FALSE;
   PARBEGIN
    PROCESSONE;
    PROCESSTWO
   PAREND
  END.
```

FIGURA: EXCLUSION MUTUA CON "TESTANDSET"

LA VARIABLE BOOLEANA ACTIVE ES VERDADERA SI CUALQUIERA DE LOS 2 PROCESOS ESTA EN SU SECCION CRITICA Y ES FALSA DE OTRO MODO. EL PROCESO UNO BASA SU DESICION PARA ENTRAR A SU SECCION CRITICA EN SU VARIABLE LOCAL BOOLEANA "ONECANNOTENTER". MIENTRAS LA PRUEBA DE ONECANNOTENTER SEA VERDADERA SE EJECUTA TESTANDSET TENIENDO COMO PARAMETROS ONECANNOTENTER Y ACTIVE. SI EL PROCESO 2 NO ESTA EN SU SECCION CRITICA, LA VARIABLE ACTIVE SERA FALSA. EL TESTANDSET ALMACENARA SU VALOR EN ONECANNOTENTER Y EL VALOR DE ACTIVE SERA VERDADERO. LA PRUEBA DEL WHILE SERA FALSA Y EL PROCESO 1 ENTRARA EN SU SECCION CRITICA. PORQUE ACTIVE HA SIDO CAMBIADA A VERDADERO, EL PROCESO 2 NO PUEDE ENTRAR A SU SECCION CRITICA.
AHORA SUPONGA QUE EL PROCESO 2 ESTA LISTO EN SU SECCION CRITICA CUANDO EL PROCESO 1 QUIERE ENTRAR. EL PROCESO 1 TIENE A LA VARIABLE ONECANNOTENTER EN VERDADERO Y ENTONCES ACTUARA REPETIDAMENTE EL PROCESO TESTANDSET. PORQUE EL PROCESO 2 ESTA EN SU SECCION CRITICA, LA VARIABLE ACTIVE ESTA PERMANENTEMENTE EN VERDADERO. CADA TESTANDSET ENCUENTRA A ACTIVE EN VERDADERO, LA PRUEBA PARA ONECANNOTENTER VERDADERA Y LA PRUEBA PARA ACTIVE VERDADERA. SIN EMBARGO, EL

PROCESO 1 CONTINUA SU ESPERA HASTA QUE EL PROCESO 2 EVENTUALMENTE SE LIBERE DE SU SECCION CRITICA Y LA PRUEBA PARA ACTIVE SEA FALSA. EN ESTE PUNTO EL TESTANDSET ENCONTRARA A ACTIVE FALSA (Y SU PRUEBA VERDADERA PARA LA UNICA SALIDA DEL PROCESO 2) Y LA PRUEBA PARA ONECANNOTENTER FALSA, ENTONCES EL PROCESO 1 ENTRARA A SU SECCION CRITICA.
ESTA SOLUCION PUEDE OCURRIR DE UNA POSPOSICION INDEFINIDA, PERO ESTO ES ALTAMENTE DISGUSTABLE, ESPECIALMENTE SI AHI SON MULTIPLES PROCESOS. TAN PRONTO COMO UN PROCESO SE LIBERE DE SU SECCION CRITICA, LA PRUEBA PARA ACTIVE SERA FALSA, EL OTRO TESTANDSET DEL PROCESO ESTA COMO UN "GRAB" EN ACTIVE (PORQUE LA PRUEBA ES VERDADERA) ANTES QUE EL PRIMER PROCESO PUEDA ENTRAR EN UN "LOOP" Y LA PRUEBA PARA ACTIVE SEA VERDADERA.

4.11 SEMAFOROS

LA NOCION DE LA LLAVE DE LA ABSTRACCION DE DIJkSTRA DE LA EXCLUSION MUTUA ES UN CONCEPTO DE SEMAFOROS. UN SEMAFORO ES UNA VARIABLE DE PROTECCION CUYO VALOR PUEDE SER ACCESADO Y ALTERADO SOLAMENTE POR LA OPERACION P Y V, Y UNA OPERACION DE INICIALIZACION, NOSOTROS LA LLAMAMOS INICIALIZACION DE SEMAFORO. SEMAFOROS BINARIOS PUEDEN ASUMIR SOLAMENTE EL VALOR 0 O EL VALOR 1. SEMAFOROS CONTADORES PUEDEN ASUMIR UN VALOR ENTERO NO NEGATIVO.
LA OPERACION P SOBRE EL SEMAFORO S, SE ESCRIBE P(S) Y ES LA SIGUIENTE:

```
IF S > 0 THEN S:=S-1
  ELSE (ESPERA SOBRE S)
```

LA OPERACION V SOBRE EL SEMAFORO S, SE ESCRIBE V(S) Y ES LA SIGUIENTE:

```
IF  (UNO O MAS PROCESOS  ESTAN  ESPERANDO
     SOBRE S)
THEN (DEJA UNO DE ESTOS PROCESOS EN EJECUCION)
        ELSE S:=S+1;
```

NOSOTROS ASUMIMOS UNA DISIPLINA DE COLA DE PRIMERO-ENTRA-PRIMERO-SALE (FIFO) PARA LOS PROCESOS QUE ESTAN ESPERANDO QUE SE COMPLETE UN P(S).
COMO UN TESTANDSET, P Y V SON INDIVISIBLES. LA EXCLUSION MUTUA SOBRE UN SEMAFORO S, ESTA REFORZADO CON P(S) Y V(S). SI VARIOS PROCESOS ATACAN UN P(S) SIMULTANEAMENTE, SOLAMENTE UNO SERA ALOJADO PARA SER PROCESADO, LOS DEMAS TENDRAN QUE ESPERAR.
LAS OPERACIONES A SEMAFOROS PUEDEN SER IMPLEMENTADAS EN SOFTWARE Y HARDWARE. ELLOS SON COMUNMENTE IMPLEMENTADAS EN LOS NUCLEOS DE LOS SISTEMAS OPERATIVOS DONDE EL ESTADO DEL PROCESO DEL SWITCHEO ES CONTROLADO.

LA SIGUIENTE FIGURA MUESTRA COMO LOS SEMAFOROS PUEDEN SER USADOS PARA REFORZAR LA EXCLUSION MUTUA. AQUI P(ACTIVE) ES EQUIVALENTE A "ENTERMUTUALEXCLUSION" Y V(ACTIVE) ES EQUIVALENTE A "EXITMUTUALEXCLUSION".

```
PROGRAM SEMAPHORE EXAMPLEONE;
VAR ACTIVE:SEMAPHORE;
PROCEDURE PROCESSONE;
BEGIN
 WHILE TRUE DO
 BEGIN
  PRELIMINARYSTUFFONE;
  P(ACTIVE);
  CRITICALSECTIONONE;
  V(ACTIVE);
  OTHERSTUFFONE
 END
END;
PROCEDURE PROCESSTWO;
BEGIN
 WHILE TRUE DO
 BEGIN
  PRELIMINARYSTUFFTWO;
  P(ACTIVE);
  CRITICALSECTIONTWO;
  V(ACTIVE);
  OTHERSTUFFTWO
 END
END;
BEGIN
 SEMAPHOREINITIALIZE (ACTIVE,1);
 PARBEGIN
  PROCESSONE;
  PROCESSTWO
 PAREND
END.
```

FIGURA: PROCESO DE SINCRONIZACION DE SEMAFOROS.

4.12 PROCESO DE SINCRONIZACION DE SEMAFOROS

CUANDO EL PROCEDIMIENTO USA UN REQUERIMIENTO DE I/O, ALGUNOS OTROS PROCESOS PUEDEN DESPERTAR DEL BLOQUEO. TAL COMO UNA INTERACCION ES EL EJEMPLO DEL PROTOCOLO "BLOOCKAUTA ALGUNOS PRELIMINARYSTUFFONE Y ENTONCES EJECUTA P(EVENT OF INTEREST). EL SEMAFORO HA SIDO INICIALIZADO EN CERO ASI QUE EL PROCESO DEBE ESPERAR. EVENTUALMENTE EL PROCESO 2 EJECUTA V(EVENT OF INTEREST) PARA SENALAR QUE EL EVENTO HA OCURRIDO. ESTO ALOJA AL PROCESO UNO PARA PROCESARLO.
NOTE QUE ESTE MECANISMO TRABAJA AUNQUE EL PROCESO 2 DETECTE

Y SENALE EL EVENTO ANTES QUE EL PROCESO UNO EJECUTE P(EVENT OF INTEREST), ELUTA ALGUNOS PRELIMINARYSTUFFONE Y ENTONCES EJECUTA P(EVENT OF INTEREST).
EL SEMAFORO HA SIDO INICIALIZADO EN CERO ASI QUE EL PROCESO DEBE ESPERAR. EVENTUALMENTE EL PROCESO 2 EJECUTA V(EVENT OF INTEREST) PARA SENALAR QUE EL EVENTO HA OCURRIDO. ESTO ALOJA AL PROCESO UNO PARA PROCESARLO.
NOTE QUE ESTE MECANISMO TRABAJA AUNQUE EL PROCESO 2 DETECTE Y SENALE EL EVENTO ANTES QUE EL PROCESO UNO EJECUTE P(EVENT OF INTEREST), EL SEMAFORO TENDRA QUE SER INCREMENTADO DE 0 A 1, ASI QUE P(EVENT OF INTEREST) SERA SIMFLEMENTE DECREMENTANDO EL SEMAFORO DE 1 A 0 Y EL PROCESO UNO SERA PROCESADO SIN QUE ESPERE EL EVENTO.

```
PROGRAM BLOCKANDWAKEUP;
VAR EVENT OF INTEREST:SEMAPHORE;
PROCEDURE PROCESSONE;
BEGIN
 PRELIMINARYSTUFFONE;
 P(EVENT OF INTEREST);
 OTHERSTUFFONE
END;
PROCEDURE PROCESSTWO;
BEGIN
 PRELIMINARYSTUFFTWO;
 V(EVENT OF INTEREST);
 OTHERSTUFFTWO
END;
BEGIN
 SEMAPHORE INICIALIZE(EVENT OF INTEREST, 0);
 PARBEGIN
  PROCESSONE;
  PROCESSTWO
 PAREND
END.
```

FIGURA:BLOCK/WAKEUP PROCESO SINCRONIZADO CON SEMAFOROS.

4.13 LA RELACION PRODUCTOR-CONSUMIDOR

EN UN PROGRAMA SECUENCIAL, DONDE UN PROCEDIMIENTO LLAMA A OTRO Y SE PASAN DATOS, LOS PROCEDIMIENTOS SON PARTE DE UN PROCESO SENCILLO, ELLOS NO HACEN OPERACIONES CONCURRENTES.
PERO DONDE UN PROCESO PASA DATOS A OTRO PROCESO, EL PROBLEMA ES MUCHO MAS COMPLEJO. COMO ESTA TRANSMICION ES UN EJEMPLO ENTRE PROCESOS CONSIDERE LA SIGUIENTE RELACION PRODUCTOR CONSUMIDOR. SUPONGA QUE UN PROCESO, UN PRODUCTOR ESTA GENERANDO INFORMACION QUE UN SEGUNDO PROCESO UN CONSUMIDOR ESTA USANDO. SUPONGA QUE ELLOS SE COMUNICAN POR MEDIO DE UNA

VARIABLE COMPARTIDA ENTERA, "NUMBER BUFFER". EL PRODUCTOR HACE ALGUNOS CALCULOS Y ENTONCES ESCRIBE EL RESULTADO EN NUMBER BUFFER, EL CONSUMIDOR LEE EL DATO DE NUMBER BUFFER Y LO IMPRIME.
SI CADA VEZ EL PRODUCTOR DEPOSITA UN RESULTADO EN NUMBER BUFFER EL CONSUMIDOR INMEDIATAMENTE LO LEE Y LO IMPRIME, ENTONCES LA IMPRESION DE SALIDA REPRESENTA FIELMENTE LA CORRIDA DE NUMEROS GENERADOS POR EL PRODUCTOR.
PERO SUPONGAMOS QUE LA VELOCIDAD DE LOS PROCESOS SON DE DISTINTO TIPO. SI EL CONSUMIDOR ESTA OPERANDO MAS RAPIDO QUE EL PRODUCTOR, EL CONSUMIDOR LEERA E IMPRIMIRA EL MISMO NUMERO VARIAS VECES ANTES QUE EL PRODUCTOR DEPOSITE EL SIGUIENTE NUMERO. SI EL PRODUCTOR ESTA OPRANDO MAS RAPIDO QUE EL CONSUMIDOR, EL PRODUCTOR ESCRIBIRA SOBRE EL RESULTADO PREVIO ANTES DE QUE EL CONSUMIDOR TENGA OPORTUNIDAD DE LEERLO E IMPRIMIRLO. UN PRODUCTOR PODRIA HACER ESTO (MUY RAPIDO) EN EL ACTO VARIAS VECES, DE TAL MANERA QUE ALGUNOS RESULTADOS SE PERDERIAN.
OBVIAMENTE, LA CONDUCTA QUE DESEAMOS AQUI PARA EL PRODUCTOR Y EL CONSUMIDOR, PARA COOPERAR DE TAL MANERA QUE LOS DATOS ESCRITOS EN NUMBER BUFFER NO SEAN PERDIDOS NI DUPLICADOS. ESFORZANDO TAL CONDUCTA ES UN EJEMPLO DE SINCRONIZACION.
LA SIGUIENTE FIGURA MUESTRA UN PROGRAMA CONCURRENTE QUE USA OPERACIONES DE SEMAFOROS PARA IMPLEMENTAR UNA RELACION PRODUCTOR-CONSUMIDOR.

```
PROGRAM PRODUCER CONSUMER RELATIONSHIP;
VAR  EXCLUSSIVEACCESS:SEMAPHORE;
     NUMBER DEPOSITED: SEMAPHORE;
     NUMBER BUFFER: INTEGER;
PROCEDURE PRODUCER PROCESS;
VAR NEXTRESULT: INTEGER;
BEGIN
 WHILE TRUE DO
 BEGIN
  CALCULATE NEXTRESULT;
  P(EXCLUSIVE ACCESS);
  NUMBER BUFFER:= NEXTRESULT;
  V(EXCLUSIVE ACCESS);
  V(NUMBER DEPOSITED)
 END
END;
PROCEDURE CONSUMER PROCESS;
VAR NEXTRESULT:INTEGER;
BEGIN
 WHILE TRUE DO
 BEGIN
  P(NUMBER DEPOSITED);
  P(EXCLUSIVE ACCESS);
```

```
    NEXTRESULT:= NUMBER BUFFER;
    V(EXCLUSIVE ACCESS);
    WRITE(NEXTRESULT)
   END
  END;
  BEGIN
   SEMAPHOREINITIALIZE(EXCLUSSIVE ACCESS, 1);
   SEMAPHOREINITIALIZE(NUMBER DEPOSITED, 0);
   PARBEGIN
    PRODUCER PROCESS;
    CONSUMER PROCESS
   PAREND
  END.
```

FIGURA: RELACION PRODUCTOR-CONSUMIDOR IMPLEMENTADA CON SEMAFOROS.

AQUI NOSOTROS TENEMOS USANDO 2 SEMAFOROS, EXCLUSIVE ACCESS ES USADO PARA ESFORZAR MUTUAMENTE LA EXCLUSIVIDAD DE ACCESO PARA LA VARIABLE COMPARTIDA Y NUMBER DEPOSITED ES USADA PARA EL PROCESO DE SINCRONIZACION.

4.14 SEMAFOROS CONTADORES

SEMAFOROS CONTADORES SON PARTICULARMENTE USADOS DONDE UN RECURSO ESTA SIENDO ALOJADO EN UNA AREA DE RECURSOS IDENTICOS.
EL SEMAFORO ES INICIALIZADO PARA EL NUMERO DE RECURSOS EN EL AREA. CADA OPERACION P DECREMENTA EL SEMAFORO EN 1 INDICANDO QUE OTRO RECURSO HA SIDO REMOVIDO DEL AREA Y EL RECURSO SERA REALOJADO EN OTRO PROCESO. SI LA OPERACION P ES ATACADA CUANDO EL SEMAFORO HA SIDO DECREMENTADO A 0, ENTONCES EL PROCESO DEBERA ESPERAR HASTA QUE UN RECURSO ESTE RETORNANDO AL AREA POR UNA OPERACION V.

4.15 IMPLEMENTANDO SEMAFOROS P Y V

DADO EL ALGORITMO DE DEKKER Y/O LA DISPONIBILIDAD DE LA INSTRUCCION DE MAQUINA TESTANDSET, ESTO NO SE DARIA DE LA IMPLEMENTACION P Y V CON OCUPADA ESPERA. PERO OCUPA ESPERAS QUE PUEDEN SER DAMINAS.
ANTERIORMENTE ESTUDIAMOS EL PROCESO DE ESTADOS CON MECANISMOS DE SWITCHEO IMPLEMENTADOS EN NUCLEOS DE UN S.O. FUE OBSERVADO QUE UN PROCESO REQUIERE UNA OPERACION I/O VOLUNTARIAMENTE CON SUS MISMOS BLOCKS DEPENDIENDO DE LA COMPLEXION DEL I/O. LOS PROCESOS BLOQUEADOS NO OCUPAN ESPERAR. EN LUGAR DE ESTO ABANDONAN EL PROCESADOR Y EL HILO DEL NUCLEO DEL PCB DEL PROCESO DENTRO DE LA LISTA DE BLOQUEADOS. EL PROCESO ASI QUEDA DORMIDO HASTA QUE SEA

DESPERTADO POR EL NUCLEO QUE REMUEVE LOS PROCESOS DE LA LISTA DE BLOQUEADOS Y LOS HILOS DE LA LISTA DE LOS READY. LAS OPERACIONES DE SEMAFOROS PUEDEN SER IMPLEMENTADAS EN EL NUCLEO PARA EVITAR QUE SE OCUPE ESPERANDO.
UN SEMAFORO ESTA IMPLEMENTADO COMO UNA VARIABLE DE PROTECCION Y UNA COLA EN LA CUAL LOS PROCESOS PUEDEN ESPERAR UNA OPERACION V. CUANDO UN PROCESO ATACA UNA OPERACION P SOBRE UN SEMAFORO QUIEN ACTUALMENTE VALE 0, EL PROCESO ABANDONA EL PROCESADOR Y SE BLOQUEA A SI MISMO PARA ESPERAR UNA OPERACION V SOBRE EL SEMAFORO. LAS SECUENCIAS DE EJECUCION DE LOS NUCLEOS DEL PCB DEL PROCESADOR DENTRO DE LA COLA DE PROCESOS ESPERANDO SOBRE ESTE SEMAFORO (NOSOTROS ASUMIMOS UNA DISIPLINA DE COLA PRIMERO-ENTRA-PRIMERO-SALE FIFO). OTRAS DISIPLINAS, INCLUYENDO LA PRIORIDAD EN LA COLA SERAN INVESTIGADAS. LOS NUCLEOS ENTONCES ASIGNAN AL PROCESADOR EL SIGUIENTE PROCESO LISTO.
EL PROCESO EN UNA COLA DE SEMAFOROS EVENTUALMENTE MUEVE LA CABEZA DE LA COLA, ENTONCES EN LA SIGUIENTE OPERACION V REMUEVE EL PROCESO DE LA COLA DE SEMAFOROS Y LO COLOCA EN LA LISTA DE READY. POR SUPUESTO QUE LOS PROCESOS ATACAN SIMULTANEAMENTE LAS OFERACIONES P Y V SOBRE UN SEMAFORO Y GARANTIZANDO EL ACCESO EXCLUSIVO PARA EL SEMAFORO POR EL NUCLEO. NOTE QUE ESTO ES UN CASO ESPECIAL DE SISTEMAS UNIPROCESADORES, LA INDIVISIBILIDAD DE P Y V PUEDE SER ASEGURADA SIMPLEMENTE DISMINUYENDO LAS INTERRUPCIONES MIENTRAS P Y V OPERACIONES ESTAN MANIPULANDO UN SEMAFORO. ESTO PREVEE AL PROCESADOR DE COMENZAR A USURPAR HASTA QUE LA MANIPULACION ESTE COMPLETA (EN LA CUAL EL PUNTO DE INTERRUPCION ESTA DE NUEVO HABILITADA).

DEADLOCK

5.1 INTRODUCCION

UN PROCESO EN UN SISTEMA DE MULTIPROGRAMACION SE DICE QUE ESTA EN UN ESTADO DE DEADLOCK SI EL ESTA ESPERANDO POR UN EVENTO EN PARTICULAR EL CUAL NO VA HA OCURRIR, SI UN SISTEMA ESTA EN DEADLOCK ENTONCES UNO O MAS PROCESOS ESTAN EN DEADLOCK; ES DECIR EL CONCEPTO DE DEADLOCK ES ESTAR ESPERANDO POR ALGO QUE VA A SUEDER PERO QUE REALMENTE NO VA A SUCEDER, ESTO ES UNA CONDICION NO PREVISTA, ASI SE PUEDE QUEDAR EN FORMA INDEFINIDA ESTE PROCESO.
EN UN SISTEMA DE MULTIPROGRAMACION EL COMPARTIR LOS RECURSOS ES UNO DE LOS PRINCIPALES OBJETIVOS DEL SISTEMA OPERATIVO. CUANDO LOS RECURSOS SON COMPARTIDOS POR UNA POBLACION DE USUARIOS CADA UNO DE ELLOS MANTIENE UN CONTROL EN EXCLUSIVA SOBRE UNO DE LOS RECURSOS EN PARTICULAR PARA ESTE USUARIO, ESTO PUEDE OCASIONAR UN DEADLOCK YA QUE LOS DEMAS USUARIOS PROBABLEMENTE ESTEN ESPERANDO EL O LOS RECURSOS QUE ESTE USUARIO ESTE OCUPANDO EN ESTE MOMENTO.
EN ESTOS PUNTOS DISCUTIMOS EL PROBLEMA DEL DEADLOCK Y SUMARIZAMOS LOS PRINCIPALES RESULTADOS DE UNA INVESTIGACION EN LAS AREAS DE SU-PREVENCION , SU-DETECCION, EL EVITARLOS Y EL COMO RECUPERARLOS.
TAMBIEN SE DISCUTIRA EL SOBRE PRECIO U OVERHEAD (SOBRE TRABAJO) QUE TENDRIA LA CORRECCION DE ESTE FENOMENO Y TAMBIEN SE CONSIDERARAN LOS BENEFICIOS QUE PUEDEN SER DERIVADOS.

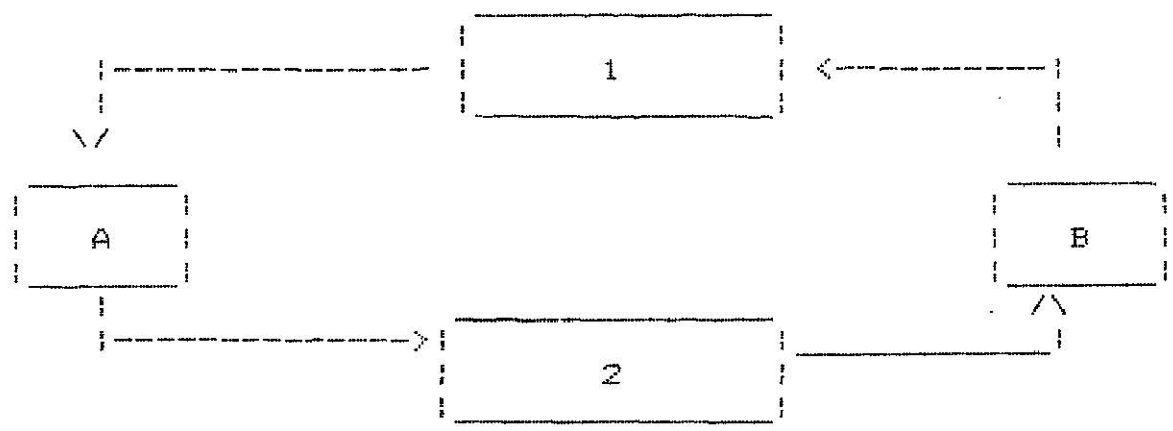
5.2 EJEMPLOS DE DEADLOCK (NUDOS)

QUIZAS LA MANERA MAS SIMPLE EN QUE SE PUEDA REPRESENTAR UN DEADLOCK ES EN UN PROGRAMA QUE SE QUEDA ESPERANDO ALGO O ALGUN RECURSO O SEA SE CICLA Y LA MANERA DE SALIRNOS DE ESTE DEADLOCK ES CANCELANDO EL PROGRAMA.
OTRO EJEMPLO SIMPLE DE UN DEADLOCK ES EL QUE OCURRE CON FRECUENCIA EN LAS GRANDES CIUDADES, ESTE ES UN PROBLEMA DE TRAFICO EN EL CUAL EXISTE UN BLOQUEO EN LA CIRCULACION, PARA ESTO TIENE QUE ACUDIR UNO O MAS AGENTES DE TRANSITO PARA TRATAR DE REESTABLECER LA CIRCULACION AUNQUE SE TENGA QUE REALIZAR UN ESFUERZO EXTRA Y UNA PERDIDA DE TIEMPO CONSIDERABLE.

5.3 UN DEADLOCK POR UN SOLO RECURSO

LOS MAYORES NUDOS QUE SON ORIGINADOS EN LOS S.O. SON DEBIDO A LA CONTENCION NORMAL DE LOS RECURSOS DEDICADOS, POR EJEMPLO LOS RECURSOS QUE PUEDEN SER USADOS POR UN USUARIO

SOLAMENTE A LA VEZ SON LLAMADOS NORMALMENTE RECURSOS EN SERIE REUSUABLES, ESTE EJEMPLO SE ILUSTRA EN LA SIGUIENTE FIGURA EN DONDE LOS RECTANGULOS SON LOS RECURSOS Y LOS CUADROS SON LOS PROCESOS, LA FLECHA QUE VA DE UN PROCESO A UN RECURSO INDICA QUE ESTE RECURSO ESTA SIENDO SOLICITADO POR EL PROCESO PERO QUE AUN NO HA SIDO CONCEDIDO Y LA FLECHA QUE VA DE UN RECURSO A UN PROCESO INDICA QUE ESTE RECURSO PERTENECE AL PROCESO EN ESTE INSTANTE.



EL DIAGRAMA ILUSTRA UN DEADLOCK EL PROCESO A RETIENE EL RECURSO B Y NECESITA EL RECURSO 2 PARA CONTINUAR MIENTRAS QUE EL PROCESO B RETIENE EL RECURSO 2 Y SOLICITA AL RECURSO 1 PARA CONTINUAR ESTO ORIGINA UN DEADLOCK POR UNA ESPERA CIRCULAR YA QUE LOS PROCESOS A Y B SE ESTAN ESPERANDO MUTUAMENTE A QUE TERMINEN PERO AMBOS SE IMPIDEN TERMINAR.

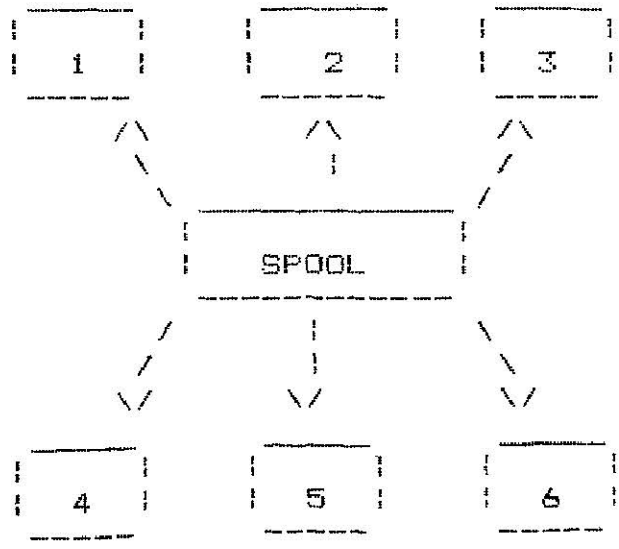
5.4 LOS DEADLOCK EN LOS SISTEMAS DE SPOOLING

LOS SISTEMAS DE SPOOLING SON TAMBIEN A MENUDO CANDIDATOS A LOS DEADLOCK, UN SISTEMA DE SPOOLING ES USADO PARA MEJORAR EL S.O. ATRAVES DE ASOCIAR LOS RECURSOS CON UNA BAJA VELOCIDAD DE OPERACION TALES COMO IMPRESORAS, LECTORAS DE TARJETAS, ETC. POR EJEMPLO SI UN PROGRAMA ESTA MANDANDO LINEAS A LA IMPRESORA DEBE ESPERAR A QUE CADA LINEA SEA IMPRESA ANTES DE QUE PUEDA TRANSMITIR LA SIGUIENTE LINEA, ESTO HARIA QUE EL PROGRAMA SE EJECUTE EN FORMA LENTA, PARA DARLE VELOCIDAD A ESTE PROGRAMA LAS LINEAS DE IMPRESION SON CANALIZADAS A TRAVES DE UN DISPOSITIVO MUCHO MAS RAPIDO COMO UN MANEJADOR DE DISCO DONDE SON ALMACENADOS EN FORMA TEMPORAL HASTA QUE PUEDAN SER IMPRESAS, A ESTO SE LE LLAMA UN SISTEMA DE SPOOLING. EN ALGUNOS SISTEMAS DE SPOOLING LA SALIDA COMPLETA DE UN PROGRAMA DEBE ESTAR DISPOBNIBLE ANTES DE QUE LA IMPRESION PUEDA COMENZAR, ESTO ES QUE NO SE MANDA LINEA POR LINEA SINO HASTA QUE NO SE ACABA DE MANDAR TODA LA IMPRESION AL SPOOLING, ENTONCES SE PUEDEN IR TERMINANDO PARCIALMENTE TRABAJOS QUE GENERAN LINEAS DE IMPRESION AL

SPOOLING LO CUAL VIENE A EVITAR LOS DEADLOCK YA QUE EN ESTE INTERVALO DE TIEMPO OTROS TRABAJOS PUEDEN SER COMPLETADOS, DE OTRA FORMA LA RECUPERACION TENDRIA QUE SER REESTABLECIENDO EL S.O. PERDIENDO TODO EL TRABAJO QUE SE HABIA HECHO HASTA ESE MOMENTO.
ESTO HACE VER QUE EN ALGUNOS SISTEMAS DE SPOOLING ESTAN ESPERANDO A QUE HALLA SUFICIENTE ESPACIO EN DISCO PARA PODER TERMINAR CON EL PROGRAMA, SI EL PROGRAMA TIENE MUCHA IMPRESION Y NO EXISTE SUFICIENTE ESPACIO EN DISCO ENTONCES ESTE PROGRAMA NO VA HA LIBERAR LOS RECURSOS HASTA QUE NO HAYA SUFICIENTE ESPACIO EN DISCO PARA MANDAR TODA SU IMPRESION, ENTONCES ESTO OCASIONA UN DEADLOCK PORQUE EL SPOOLING ES PEQUENO Y EL PROGRAMA NECESITA MAS SPOOLING Y EL PROGRAMA NO LIBERA TODOS LOS RECURSOS PORQUE NO HAY SPOOLING PARA GUARDAR TODA LA IMPRESION, PARA CONTINUAR ESTE SE TENDRIAN QUE CANCELAR OTROS TRABAJOS PARA AGRANDAR EL SPOOLING O SE TENDRIA QUE CANCELAR ESTE PROGRAMA Y SE PERDERIA TODO LO HECHO HASTA ESTE INSTANTE. POR LO TANTO UN PROGRAMADOR DE S.O. DEBE PREVEER EL CREAR UNA AREA DE SPOOLING SUFICIENTEMENTE GRANDE PARA EL TIPO DE PROCESOS QUE SE DESEAN EJECUTAR. HOY EN DIA LOS SPOOLING PERMITEN EMPEZAR LA IMPRESION ANTES DE QUE SE HAYA LLENADO EL SPOOLING DE TAL FORMA QUE NO ES NECESARIO DEJAR TANTO ESPACIO COMO LO ERA EN FORMA ANTERIOR.

5.5 LOS DEADLOCK EN EL SCHEDULING

EN MUCHOS SISTEMAS LOS RECURSOS SON GUARDADOS EN UNA AREA LLAMADA SPOOLING MIENTRAS NO HAYA LOS RECURSOS NECESARIOS PARA SER EJECUTADOS ESTO SE PUEDE ENTENDER MEJOR CON LA FIGURA SIGUIENTE:



LOS CUADROS REPiRESENTAN PROESOS EN ESPERA DE EJECUCION, EL NUMERO INTERIOR SU PRIORIDAD. EL SCHEDULING ASIGNA TIEMPO DE EJECUCION Y LOS RECURSOS PARA QUE UN PROCESO SE REALIZE, SIN EMBARGO ESTO SE MANEJA POR PRIORIDAD, ES DECIR LOS PROCESOS CON MAS ALTA PRIORIDAD (NUMERO MAS PEQUENO) SON ATENDIOS PRIMERO HASTA AGOTAR LOS RECURSOS DE CPU, DESPUES CUANDO SE LIBERA ALGUN RECURSO ESTE ES ASIGNADO AL PROCESO QUE LO SOLICITE, PERO PRIMERO AL DE MAS ALTA PRIORIDAD, DE TAL MANERA QUE EN LA FIGURA EL PROCESO CON PRIORIDAD 6 TAL VEZ NUNCA SE TERMINE DE EJECUTAR QUEDANDO ESTE EN UN DEADLOCK.

5.6 CONCEPTO DE RECURSOS

UN S.O. ES UN MANEJADOR DE RECURSOS. ES RESPONSABLE DE LA ASIGNACION DE RECURSOS DE VARIOS TIPOS. CONSIDERAMOS RECURSOS QUE SON REMOVIBLES TALES COMO EL CPU Y MEMORIA PRINCIPAL, UN PROGRAMA QUE OCUPA UN RANGO PARTICUCLAR DE LOCALIDADES EN MEMORIA PRINCIPAL QUIZA SERA REMOVIDO POR OTRO PROGRAMA. UN PROGRAMA QUE REQUIERA I/O NO PUEDE HACER USO EFECTIVO DE LA MEMORIA PRINCIPAL,DURANTE LA EJECUCION HASTA QUE I/O SE COMPLETE. EL CPU ES QUIZA EL RECURSO MAS REMOVIBLE EN UN SISTEMA COMPUTACIONAL.
SIEMPRE QUE UN PROCESO PARTICULAR ALCANCE UN PUNTO EN EL QUE EL CPU NO PUEDE SER USADO EFECTIVAMENTE, EL CONTROL DEL CPU ES REMOVIDO DESDE ESE PROCESO Y DADO A OTRO.
CIERTOS SUCESOS SON NO REMOVIBLES Y NO PUEDEN SER REMOVIDOS DESDE PROCESOS PARA LOS CUALES FUERON AIGNADOS. POR EJEMPLO LOS DRIVES, MIENTRAS PERTENECEN A UN PROCESO, NO PUEDEN SER TOMADOS DESDE ESE PROCESO PARA ASIGNARLO A OTRO.
ALGUNOS RECURSOS SON COMPARTIDOS POR PROCESOS Y OTROS SON EXCLUSIVOS A UNO SOLO. LOS DRIVES SON ASIGNADOS A PROCESOS SINGULARES, PERO ALGUNAS VECES CONTIENEN ARCHIVOS COMPARTIDOS POR VARIOS PROCESOS.
LOS PROGRAMAS Y DATOS SON RECURSOS QUE NECESITAN SER CONTROLADOS Y ASIGNADOS. EN SISTEMAS DE MULTIPROGRAMACION MUCHOS USUARIOS QUIZA QUIEREN USAR UN PROGRAMA EDITOR SIMULTANEAMENTE, ESTO CONSUME MEMORIA PRINCIPAL AL TRAER UNA COPIA SEPARADA DEL EDITOR PARA CADA PROGRAMA. AL INSTANTE, UNA COPIA DEL CODIGO ES TRAIDA Y MUCHAS COPIAS PARA LOS DATOS SON HECHAS, UNA PARA CADA USUARIO. EL CODIGO PUEDE SER USADO POR MUCHOS, PERO NO CAMBIADO. EL CODIGO QUE NO PUEDE SER CAMBIADO MIENTRAS ESTA EN USO ES LLAMADO REENTRANT. EL CODIGO QUE PUEDE SER CAMBIADO PERO ES REINICIALIZADO CADA VEZ QUE ES USADO ES LLAMADO SERIALLY REUSABLE. REENTRANT PUEDE SER ACCESADO POR PROCESOS SIMULTANEOS MIENTRAS QUE SERIALLY REUSABLE SOLO PUEDE SER USADO POR UN PROCESO A LA VEZ. CUANDO USAMOS RECURSOS PARTICULARES, DEBEMOS SER CUIDADOSOS AL ESTABLECER SIEMPRE QUE SEAN USADOS POR VARIOS PROCESOS SIMULTANEAMENTE O POR UNO A LA VEZ. LOS SEGUNDOS

SON LOS RECURSOS QUE SERAN ENVUELTOS EN DEADLOCK MAS COMUNMENTE.

5.7 CONDICIONES NECESARIAS PARA DEADLOCK

1. LOS PROCESOS PIDEN EL CONTROL EXCLUSIVO DE LOS RECURSOS QUE REQUIEREN (CONDICION DE EXCLUSION MUTUA)
2. LOS PROCESOS TOMAN RECURSOS YA ASIGNADOS A ELLOS MIENTRAS ESPERAN POR UN RECURSO ADICIONAL (CONDICION DE ESPERA)
3. LOS RECURSOS NO PUEDEN SER REMOVIDOS DESDE LOS PROCESOS PARA LOS CUALES FUERON ASIGNADOS HASTA QUE SEAN USADOS POR COMPLETO (CONDICION DE NO-PREMPTON)
4. UNA CADENA CIRCULAR DE PROCESOS EXISTENTES EN LA CUAL CADA PROCESO TOMA 1 O MAS RECURSOS QUE SON REQUERIDOS POR EL SIGUIENTE PROCESO EN LA CADENA (CONDICION DE ESPERA CIRCULAR)

5.8 INVESTIGACION DE AREAS DE DEADLOCK

HAY 4 AREAS EN LA INVESTIGACION DEL DEADLOCK:
1. PREVENIR DEADLOCK: NUESTRO INTERES ES CONDICIONAR UN SISTEMA PARA REMOVER CUALQUIER PROBABILIDAD DE QUE OCURRA DEADLOCK. PREVENIR ES UNA CLARA SOLUCION EN CUANTO A DEADLOCK CONCIERNE, PERO METODOS PREVENTIVOS PUEDEN RESULTAR UN RECURSO POBREMENTE UTILIZADO. SIN EMBARGO; SON AMPLIAMENTE PRACTICADOS.
2. EVITAR DEADLOCK: EL OBJETIVO ES IMPLANTAR MENOS CONDICIONES RESTRINGENTES QUE EN PREVENCION PARA OBTENER MEJOR UTILIZACION DE RECURSOS. EVITAR NO PRECONDICIONA AL SISTEMA A REMOVER TODA POSIBILIDAD DE DEADLOCK. SIEMPRE QUE UN DEADLOCK ES DETECTADO ES CUIDADOSAMENTE PUESTO A UN LADO.
3. DETECCION DE DEADLOCK: METODOS SON USADOS EN SISTEMAS QUE ALOJAN DEADLOCKS POR OCURRIR, SE QUIERA O NO. EL OBJETIVO ES DETERMINAR SI UN DEADLOCK A OCURRIDO Y DETERMINAR AQUELLOS RECURSOS Y PROCESOS ENVUELTOS EN EL. UNA VEZ DETERMINADO ESTO, PUEDE EL DEADLOCK SER QUITADO DEL SISTEMA.
4. RECUPERACION DE DEADLOCK: METODOS USADOS PARA QUITAR DEADLOCK DE UN SISTEMA PARA PODER OPERAR LIBRE UN PROCESO Y PODER LIBERAR SUS RECURSOS. UNA VEZ RECUPERADOS LOS PROCESOS ACCESADOS SON REESTABLECIDOS NORMALMENTE DESDE EL PRINCIPIO.

5.9 PREVENCION DE DEADLOCK

EL MAS FRECUENTE USO DE DISEÑADORES QUE TRATAN EL PROBLEMA DE DEADLOCK ES PREVENIRLO.
HAVENDER CONCLUYE QUE SI CUALQUIERA DE LAS CONDICIONES NECESARIAS ES NEGADA, ES IMPOSIBLE QUE OCURRA UN DEADLOCK. SUGIERE LAS SIGUIENTES ESTRATEGIAS DE NEGAR VARIAS CONDICIONES NECESARIAS.

- CADA PROCESO QUE REQUIERE RECURSO, NO PUEDE PROCEDER HASTA QUE LE HAYAN SIDO ASIGNADOS TODOS.
- SI UN PROCESO REQUIERE CIERTOS RECURSOS Y LE SON NEGADOS, LIBERA LOS QUE TIENE Y LOS VUELVE A PEDIR JUNTO CON LOS QUE NECESITA.
- IMPOSICION DE UN ORDEN LINEAL EN LOS RECURSOS PARA TODOS LOS PROCESOS

5.10 NEGANDO LA CONDICION EN ESPERA POR (WAIT FOR)

LA PRIMER ESTRATEGIA REQUIERE QUE TODOS LOS RECURSOS QUE UN PROCESO NECESITARA SEAN REQUERIDOS A LA VEZ, EL SISTEMA LOS CONCEDE CON LA BASE DE 'TODO O NADA'. SI EL CONJUNTO DE RECURSOS NECESITADOS POR UN PROCESO ESTA DISPONIBLE, EL SISTEMA SE LOS CONCEDE A UN TIEMPO Y LOS ALOJA PARA EMPEZAR A TRABAJAR. SI NO ESTA DISPONIBLE, EL PROCESO TIENE QUE ESPERAR HASTA QUE SE COMPLETE EL CONJUNTO DE DISPONIBLE. MIENTRAS ESPERA, NO TIENE NINGUN RECURSO.
ESTO SUENA BIEN, PERO PUEDE CONDUCIR A SERIOS CONSUMOS DE RECURSOS.
UN ENFOQUE FRECUENTEMENTE USADO POR SISTEMAS, PARA OBTENER MEJOR UTILIZACION DE RECURSOS EN ESTAS CIRCUNSTANCIAS, ES DIVIDIR UN PROGRAMA EN VARIOS QUE CORRAN RELATIVAMENTE INDEPENDIENTES UNO DEL OTRO, ENTONCES EL ALOJO DE RECURSOS PUEDE SER CONTROLADO POR PASOS MEJOR QUE POR EL PROCESO ENTERO. ESTO REDUCE EL CONSUMO PERO ENVUELVE UN MAYOR OVERHEAD.
ESTA ESTRATEGIA PUEDE CAUSAR POSPOSICION INDEFINIDA CUANDO NO TODOS LOS RECURSOS REQUERIDOS ESTEN DISPONIBLES A LA VEZ. EL SISTEMA PUEDE ALOJAR UN NUMERO SUFICIENTE DE JOBS PARA SER CONPLEMENTADOS Y LIBERAR SUS RECURSOS ANTES QUE UN JOB QUE ESPERA SE PUEDA EJECUTAR. MIENTRAS LOS RECURSOS REQUERIDOS ESTAN SIENDO ACUMULADOS NO PUEDEN SER USADOS POR OTRO JOB Y LOS RECURSOS SON DESPERDICIADOS. HAY CONTROVERSIAS SOBRE AQUIEN DAR ESOS RECURSOS SIN USO.

5.11 NEGANDO LA CONDICION NO APROPIATIVIDAD (NO PREEMPTION)

SUPONE QUE UN SISTEMA ALOJA PROCESOS QUE TIENEN RECURSOS MIENTRAS ESPERA RECURSOS ADICIONALES. MIENTRAS PERMANEZCAN DISPONIBLES SUFICIENTES RECURSOS PARA SATISFACER TODO LOS REQUERIMIENTOS NO HABRA DEADLOCK.PERO CONSIDERAMOS QUE PASA SI UN REQUERIMIENTO NO HA SIDO SATISFECHO. AHORA UN PROCESO TIENE RECURSOS QUE UN SEGUNDO PROCESO PUEDE NECESITAR PARA PODER SEGUIR MIENTRAS QUE UN SEGUNDO PROCESO PUEDE NECESITAR PARA PODER SEGUIR MIENTRAS QUE EL SEGUNDO PUEDE TENER LOS RECURSOS NECESITADOS POR EL PRIMER PROCESO. ESTO ES DEADLOCK.

LA SEGUNDA ESTRATEGIA DE HAVENDER REQUIERE QUE CUANDO UN PROCESO TENGA PROCESOS Y LE SEAN NEGADOS RECURSOS ADICIONALES, LIBERE LOS RECURSOS QUE TIENE Y SI ES NECESARIO, LOS PIDA JUNTO CON LOS ADICIONALES. LA IMPLEMENTACION DE ESTA ESTRATEGIA EFECTIVAMENTE NIEGA LA CONDICION DE "NO-PREEMTTON". LOS RECURSOS PUEDEN SER REMOVIDOS DESDE LOS PROCESOS QUE LOS TIENEN ANTES QUE SE COMPLETEN ESOS PROCESOS.
AQUI TAMBIEN, EL SIGNIFICADO DE PREVENCION NO ES COSTEABLE. UN PROCESO QUE USA CIERTOS RECURSOS EN ALGUN TIEMPO, CUANDO LIBERA ESOS RECURSOS PUEDE PERDER TODO SU TRABAJO HASTA ESE PUNTO. ESTO QUIZA PUEDE SER UN ALTO PRECIO POR PAGAR. LA PREGUNTA ES, QUE TAN FRECUENTE OCURRE ESTO ? SI NO OCURRE FRECUENTEMENTE EL COSTO ES RELATIVAMENTE BAJO. SI DE OTRA MANERA OCURRE CON FRECUENCIA, EL COSTO ES SUBSTANCIAL Y LOS RESULTADOS INTERRUMPIDOS, PARTICULARMENTE CUANDO PROCESOS DE ALTA PRIORIDAD NO PUEDEN SER COMPLETADOS A UN TIEMPO.
UNA SERIA CONSECUENCIA DE ESTA ESTRATEGIA ES LA POSIBILIDAD DE UNA POSPOSICION INDEFINIDA. UN PROCESO PUEDE INDEFINIDA Y REPETIDAMENTE REQUERIR Y LIBERAR LOS MISMOS RECURSOS. SI ESTO OCURRE, EL SISTEMA NECESITARA REMOVER EL PROCESO PARA QUE OTRO PROCESO PUEDA SER EJECUTADO. LA POSIBILIDAD DE UNA POSPOSICION INDEFINIDA PODRIA NO SER MUY VISIBLE PARA EL SISTEMA, PERO NO PUEDE SER IGNORADO. EN ESTE CASO, EL PROCESO CONSUME RECURSOS SUBSTANCIALES Y DEGRADA LA EJECUCION DEL SISTEMA.

5.12 NEGANDO LA CONDICION ESPERA CIRULAR (WAIT-CIRCULAR)

LA TERCER ESTRATEGIA DE HAVENDER NIEGA LA POSIBILIDAD DE UNA ESPERA CIRCULAR. PORQUE TODOS LOS RECURSOS SON NUMERADOS Y LOS PROCESOS VAN A REQUERIRLOS EN ORDEN ASCENDENTE, ESTO LO HACE IMPOSIBLE.
ESTA ESTRATEGIA HA SIDO IMPLANTADA POR ALGUNOS SISTEMAS, PERO NO SIN DIFICULTADES, SI UN NUEVO RECURSO ES SUMADO A LA INSTALACION, ALGUNOS PROGRAMAS Y SISTEMAS TIENEN QUE SER RESCRITOS.
- CLARAMENTE, CUANDO LOS NUMEROS DE RECURSOS SON ASIGNADOS REFLEJAN EL ORDEN NORMAL EN EL CUAL LOS JOBS LOS UTILIZAN. PARA ELLOS, LA OPERACION EFICIENTE ES DE ESPERARSE. PERO PARA JOBS QUE NECESITAN LOS RECURSOS EN DIFERENTE ORDEN EN QUE FUERON ASUMIDOS POR LA INSTALACION, LOS RECURSOS PUEDEN SER TOMADOS, POSIBLEMENTE ANTES DE QUE SEAN USADOS. ESTO RESULTA UN DESPERDICIO PORQUE SON TOMADOS PERO NO USADOS.
- UNO DE LOS MAS IMPORTANTES OBJETIVOS DE LOS S.O. ES LA CREACION DE MEDIOS AMBIENTES AMIGABLES, LOS USUARIOS PUEDEN DESARROLLAR SUS APLICACIONES CON UN MINIMO DE CONTAMINACION DE SUS MEDIOS POR HARDWARE Y RESTRICCIONES DE SOFTWARE. EL ORDEN LINEAL DE HAVENDER REALMENTE ELIMINA LA

POSIBILIDAD DE ESPERA CIRCULAR, SIN EMBARGO ES UN EFECTO NEGATIVO SOBRE LAS HABILIDADES DEL USUARIO PARA ESCRIBIR LIBRE Y FACILMENTE CODIGOS DE APLICACIONES.

5.13 ALGORITMO DEL BANQUERO

EVITAR DEADLOCK SIENDO CUIDADOSO CUANDO LOS RECURSOS SON SENALADOS. QUIZAS EL MAS FAMOSO ALGORITMO PARA EVITAR DEADLOCK ES EL ALGORITMO DEL BANQUERO DE DIJKSTRA, LLAMADO POR ESTE INTERESANTE NOMBRE PORQUE INVOLUCRA A UN BANQUERO QUIEN HACE PRESTAMOS Y RECIBE PAGOS DE RECURSOS RECIBIDOS DE CAPITAL. NOSOTROS PAREAFRASEAMOS EL ALGORITMO AQUI EN EL CONTEXTO DE SISTEMAS OPERATIVOS DE RECURSOS COLOCADOS.

5.14 EL ALGORITMO DEL BANQUERO DE DIJISTRA

CUANDO NOSOTROS NOS REFERIMOS A LOS RECURSOS EN LO SIGUIENTE QUEREMOS DECIR RECURSOS DEL MISMO TIPO. EL ALGORITMO DEL BANQUERO ES FACILMENTE EXTENDIBLE PARA LA ACUMULACION DE RECURSOS DE DIVERSOS DIFERENTES TIPOS, POR EJEMPLO CONSIDERE LA COLOCACION DE UNA CANTIDAD DE IDENTICOS DRIVES DE CINTAS. UN S.O. COMPARTE UN NUMERO FIJO EQUIVALENTE DE DRIVES DE CINTA, ENTRE UN NUMERO FIJO DE USUARIOS. CADA USUARIO ESPECIFICA POR ANTICIPADO EL NUMERO MAXIMO DE DRIVES DE CINTAS QUE EL O ELLA NECESITARAN DURANTE LA EJECUCION DE JOBS (TRABAJOS) EN EL SISTEMA. EL S.O. ACEPTARA UN REQUERIMIENTO DE USUARIO SI LA NECESIDAD MAXIMA DE DRIVES DE CINTA NO EXEDIO. UN USUARIO PUEDE OBTENER O LIBERAR UN DRIVE DE CINTAS UNO POR UNO. ALGUNAS VECES UN USUARIO TIENE QUE ESPERAR PARA OBTENER UN DRIVE DE CINTA ADICIONAL, PERO EL S.O. GARANTIZA UNA ESPERA FINITA. EL NUMERO NORMAL DE RECURSOS ASIGNADOS PARA UN USUARIO NUNCA EXCEDERA DE LA MAXIMA NECESIDAD ESTABLECIDA. SI EL S.O. ES CAPAZ DE SATISFACER LA MAXIMA NECESIDAD DE LOS RECURSOS DEL USUARIO, ENTONCES EL USUARIO GARANTIZA QUE LOS RECURSOS SERAN USADAS Y LIBERADAS DURANTE EL TIEMPO FINITO. EL ESTADO NORMAL DEL SISTEMA ES LLAMADO LIBRE O SEGURO SI ES POSIBLE PARA EL S.O. PERMITIR A TODOS LOS USUARIOS NORMALES COMPLETAR SUS JOBS DENTRO DEL TIEMPO FINITO (DE NUEVO SE ASUME QUE LOS RECURSOS SON LOS UNICOS RECURSOS REQUERIDOS POR LOS USUARIOS) SI NO, ENTONCES ES EL SISTEMA NORMAL ESTABLECIDO ES LLAMADO INSEGURO. EL ALGORITMO DEL BANQUERO DE DIJKSTRA DICE, PARA ASIGNAR LOS RECURSOS A LOS USUARIOS SOLAMENTE CUANDO LAS ASIGNACIONES RESULTEN EN ESTADOS DE SEGURIDAD MAS BIEN QUE EN ESTADO INSEGURO. UN ESTADO SEGURO ES UNO EN EL CUAL LA SITUACION DE RECURSOS ES TAL QUE TODOS LOS USUARIOS EVENTUALMENTE SERAN CAPACES DE TERMINAR Y UN ESTADO INSEGURO ES UNO QUE PODRIA EVENTUALMENTE CONDUCIR AL DEADLOCK.

5.15 EJEMPLO DE UN ESTADO SEGURO

SUPONGAMOS QUE UN SISTEMA TIENE 12 RECURSOS EQUIVALENTES Y 3 USUARIOS SE REPARTEN LOS RECURSOS COMO EN LA FIGURA SIGUIENTE:

ESTADO 1

	RECURSOS ASIGNADOS	NECESIDAD MAXIMA
USUARIO (1)	1	4
USUARIO (2)	4	6
USUARIO (3)	5	8
DISPONIBLES	2	

ESTE ESTADO ES SEGURO PORQUE ES AUN POSIBLE PARA LOS 3 USUARIOS TERMINAR. NOTE QUE EL USUARIO 2 NORMALMENTE TIENE 4 RECURSOS PARA EL Y EVENTUALMENTE NECESITARA UN MAXIMO DE 6. EL SISTEMA TIENE 12 DRIVES DE LOS CUALES 10 ESTAN NORMALMENTE EN USO Y 2 ESTAN DISPONIBLES. SI ESTOS 2 DRIVES DISPONIBLES SON DADOS AL USUARIO COMPLETANDO SU MAXIMA NECESIDAD, ENTONCES EL USUARIO 2 PUEDE TERMINAR. EL USUARIO 2 AL TERMINO LIBERA TODOS LOS RECURSOS QUE EL SISTEMA PUEDE ASIGNAR AL USUARIO 1 Y 3. EL USUARIO 1 TIENE UN RECURSOS Y NECESITA 3 MAS EVENTUALMENTE. EL USUARIO 3 TIENE 5 Y NECESITA 3 MAS EVENTUALMENTE. SI EL USUARIO 2 DEVUELVE 6, ENTONCES 3 PUEDEN SER DADOS AL USUARIO 1 QUIEN PUEDE ENTONCES TERMINAR Y REGRESAR 4 RECURSOS PARA EL SISTEMA. EL SISTEMA PUEDE ENTONCES DARLE 3 DRIVES AL USUARIO 3 QUIEN AHORA PUEDE TAMBIEN TERMINAR ASI LA CLAVE PARA EL ESTADO SEGURO ES QUE HAYA UNA MANERA PARA QUE TODOS LOS USUARIOS TERMINEN.

5.16 EJEMPLO DE UN ESTADO INSEGURO

ESTADO II

	RECURSOS ASIGNADOS	NECESIDAD MAXIMA
USUARIO (1)	8	10
USUARIO (2)	2	5
USUARIO (3)	1	3
DISPONIBLES	1	

AQUI 11 DE LOS 12 RECURSOS DEL SISTEMA ESTAN NORMALMENTE EN USO Y SOLAMENTE UNO ESTA DISPONIBLE PARA ASIGNAR EN ESTA SITUACION, NO IMPORTA CUAL USUARIO REQUIERE DEL RECURSO DESIGNABLE, NO PODEMOS GARANTIZAR QUE LOS 3 USUARIOS TERMINEN. DE HECHO SUPONGA QUE EL USUARIO 1 REQUIERE Y SE LE CONCEDE EL ULTIMO RECURSO DISPONIBLE, 3 MANERAS DE DEADLOCK PODRIAN OCURRIR SI DE VERDAD EL PROCESO NECESITA REQUERIR AL MENOS UN DRIVE MAS ANTES DE LIBERAR ALGUN DRIVE DEL CONJUNTO.
ESTO ES IMPORTANETE DE TOMAR EN CUENTA, QUE UN ESTADO INSEGURO NO IMPLICA LA EXISTENCIA O AUN LA EXISTENCIA EVENTUAL DEL DEADLOCK.
LO QUE UN ESTADO INSEGURO IMPLICA ES SIMPLEMENTE QUE ALGUNA INFORTUNADA SECUENCIA DE EVENTOS PODRIA CONDUCIR AL DEADLOCK.

5.17 EJEMPLO DE LA TRANSICION DE UN ESTADO SEGURO A UNO INSEGURO

QUE UN ESTADO SEA CONOCIDO COMO SEGURO NO IMPLICA QUE TODOS LOS ESTADOS EN EL FUTURO SEAN SEGUROS. NUESTRA POLITICA DE RECURSOS ASIGNADOS SE DEBERA CONSIDERAR CUIDADOSAMENTE.

ESTADO III

	RECURSOS ASIGNADOS	NECESIDAD MAXIMA
USUARIO (1)	1	4
USUARIO (2)	4	6
USUARIO (3)	5	8
DISPONIBLES	2	

NOTA: AHORA SUPONGA QUE EL USUARIO 3 REQUIERE DE UN RECURSO ADICIONAL. SI EL SISTEMA FUERA A CONCEDER ESTE REQUERIMIENTO, ENTONCES EL NUEVO ESTADO SERA ESTADO 4.

ESTADO IV

	RECURSOS ASIGNADOS	NECESIDAD MAXIMA
USUARIO (1)	1	4
USUARIO (2)	4	6
USUARIO (3)	6	8
DISPONIBLES	1	

CIERTAMENTE, EL ESTADE 4 NO ES NECESARIAMENTE DEADLOCK. PERO EL ESTADO SE HA IDO DE UN SEGURO A UN INSEGURO. EL ESTADO 4 CARACTERIZA UN SISTEMA EN EL CUAL LA TERMINACION DE LOS PROCESOS DE TODOS LOS USUARIOS NO PUEDE SER GARANTIZADA. UN RECURSO ESTA DISPONIBLE. UN MINIMO DE 2 RECURSOS DEBEN ESTAR DISPONIBLES PARA ASEGURAR QUE TANTO EL USUARIO 2 O EL 3 PUDIERA TERMINAR, DEVOLVIENDO SUS RECURSOS AL SISTEMA Y FINALMENTE PERMITIENDO A TODOS LOS USUARIOS TERMINAR.

5.18 ALGORITMO DE DESIGNACION DE RECURSOS DE BANQUEROS

AHORA DEBERIA ESTAR CLARO COMO LA ASIGNACION DE RECURSOS OPERAN BAJO EL ALGORITMO DEL BANQUERO DE DIJKSTRA. LAS CONDICIONES: DE EXCLUSION MUTUA, LA ESPERA Y NO PRIORIDAD SON PERMITIDAS.
LOS PROCESOS RECLAMAN USO EXCLUSIVO DE LOS RECURSOS QUE ELLOS REQUIEREN. A LOS PROCESOS LE SON PERMITIDOS RETENER RECURSOS MIENTRAS REQUIERA Y ESPERA POR RECURSOS ADICIONALES, LOS RECURSOS PUEDEN NO SER PRIORITARIOS DE UN PROCESO DE RETENCION DE ESOS RECURSOS. ES UN SISTEMA PARA REQUERIR RECURSOS DE UNO A LA VEZ. EL SISTEMA PUEDE GARANTIZAR O NEGAR CADA REQUERIMIENTO. SI EL REQUERIMIENTO ES NEGADO EL USUARIO RETIENE CUALQUIER RECURSO DESIGNADO Y ESPERA POR TIEMPO FINITO HASTA QUE EL REQUERIMIENTO ES EVENTUALOMENTE CONCEDIDO. EL SISTEMA CONCEDE REQUERIMIENTOS QUE RESULTAN EN ESTADOS SEGUROS SOLAMENTE. LOS REQUERIMIENTOS DE UN USUARIO QUE RESULTEN EN UN ESTADO INSEGURO SON NEGADOS HASTA QUE ESTE PUEDA SER EVENTUALMENTE SATISFECHO. POR SUPUESTO, PORQUE EL SISTEMA SIEMPRE ES MANTENIDO EN UN ESTADO SEGURO, MAS PRONTO O MAS TARDE (EN UN TIEMPO FINITO) TODOS LOS REQUERIMIENTOS PUEDEN SER SATISFECHOS Y TODOS LOS USUARIOS PUEDEN TERMINAR.

5.19 DEBILIDADES EN EL ALGORITMO DEL BANQUERO

EL ALGORITMO DEL BANQUERO ES DE INTERES PARA NOSOTROS PORQUE PROVEE UNA MANERA DE ASIGNACION DE RECURSOS PARA EVITAR EL DEADLOCK. ELLO PERMITE A LOS JOB'S PROCEDER EN SITUACIONES DE PREVENCION DE UN DEADLOCK. HABRIA TENIDO QUE ESPERAR. PERO EL ALGORITMO CONTIENE UN NUMERO DE DEBILIDADES SERIAS QUE PODRIAN CAUSAR A UN DISEMADOR ESCOGER OTRO ACERCAMIENTO AL PROBLEMA DEL DEADLOCK. EL ALGORITMO REQUIERE QUE HAYA UN NUMERO FIJO DE RECURSOS PARA DESIGNAR. LOS RECURSOS FRECUENTEMENTE REQUIEREN DE SERVICIOS POR CAIDAS O MANTENIMIENTO PREVENTIVO, NOSOTROS NO PODEMOS CONTAR CON EL NUMERO DE RECURSOS QUE PERMANECERAN CONSTANTEMENTE FIJOS. EL ALGORITMO REQUIERE QUE LA POBLACION DE USUARIOS PERMANEZCA FIJOS ESTO TAMBIEN ES IRRAZONABLE. LOS SISTEMAS MULTIPROGRAMADOS ACTUALES ESTAN CONSTANTEMENTE CAMBIANDO. EN

UN GRAN SISTEMA DE TIEMPO COMPARTIDO. POR EJEMPLO, ESTO NO ES USUAL, DAR SERVICIO A MAS DE 100 USUARIOS SIMULTANEAMENTE. PERO LA POBLACION DE USUARIOS CAMBIA CONSTANTEMENTE, QUIZAS TAN AMENUDO COMO CADA POCOS SEGUNDOS. EL ALGORITMO REQUIERE QUE LOS BANQUEROS GARANTIZEN QUE TODOS LOS REQUERIMIENTOS SEAN CONCEDIDOS DENTRO DEL TIEMPO FINITO. CLARAMENTE, MUCHAS GARANTIAS MEJORES SERAN NECESITADAS EN SISTEMAS REALES.
EL ALGORITMO REQUIERE QUE LOS USUARIOS ESTABLEZCAN SUS MAXIMAS NECESIDADES POR ANTICIPADO. CON DESIGNACION DE RECURSOS LLEGANDO A RESULTAR UN CRECIMIENTO DINAMICO. CON ESTO RESULTA MAS DIFICIL SABER LA NECESIDAD MAXIMA DE LOS USUARIOS, DE HECHO COMO LOS SISTEMAS PROVEEN MAS INDEPENDENCIA ENTRE USUARIOS QUE NO TIENEN LA MAS LIGERA IDEA DE LOS RECURSOS QUE NECESITAN.

5.20 DETECCION DEL DEADLOCK

LA DETECCION DEL DEADLOCK ES EL PROCESO DE DETERMINAR QUE EXISTE UN DEADLOCK Y LA IDENTIFICACION DE LOS PROCESOS Y LOS RECURSOS INVOLUCRADOS EN EL DEADLOCK. LOS ALGORITMOS DE DETECCION DEL DEADLOCK SON GENERALMENTE USADOS EN SISTEMAS EN LOS CUALES LAS PRIMERAS 3 CONDICIONES NECESARIAS PARA QUE EL DEADLOCK OCURRA SEA PERMITIDO. ESTOS ALGORITMOS ENTONCES DETERMINARAN SI UNA ESPERA CIRCULAR EXISTE.
EL USO DEL ALGORITMO DE DETECCION DEL DEADLOCK INVOLUCRA CIERTO TIEMPO CORRIDO DE OVERHEAD ASI NOSOTROS OTRA VEZ DE NUEVO HACEMOS FRENTE A LAS CONDICIONES DE INTERCAMBIO ASI PREVALECE EN S.O.

5.21 REDUCCION DE RECURSOS DE GRAFOS SENALADOS

LA DETECCION DEL DEADLOCK ES DE INTERES PARA LA DETERMINACION DE QUE UN DEADLOCK EXISTE. UNA TECNICA PARA DETECTAR DEADLOCKS IMPLICA LA REDUCCION DE GRAFOS EN EL CUAL EL PROCESO SERA COMPLETO Y EL PROCESO DE PERMANENCIA DE DEADLOCK SERA DETERMINADO.
SI UN RECURSO DE PROCESO SOLICITADO ES PRIVILEGIADO, ENTONCES DIREMOS QUE UN GRAFO PUEDE SER REDUCIDO POR TAL PROCESO. ESTA REDUCCION ES EQUIVALENTE PARA MOSTRAR LOS GRAFOS, SI ESTE PROCESO ES ADMITIDO PARA COMPLETAR Y RETORNAR ESOS RECURSOS AL SISTEMA. LA REDUCCION DE UN GRAFO POR UN PROCESO PARTICULAR ES MOSTRADO PARA MOVER LAS FLECHAS A AQUEL PROCESO DE SENALAMIENTO (RECURSOS SENALADOS A CADA PROCESO) Y POR EL REMOVIMIENTO DE FLECHAS DESDE AQUEL PROCESO A LOS RECURSOS (COMUNMENTE RECURSOS SOLICITADOS PARA AQUEL PROCESO). SI UN GRAFO PUEDE SER REDUCIDO POR TODOS ESOS PROCESOS ENTONCES NO HAY DEADLOCK. SI UN GRAFO NO PUEDE SER REDUCIDO POR TODOS ESOS PROCESOS ENTONCES LOS PROCESOS

"IRREDUCIBLES" CONSTITUYEN EL CONJUNTO DE PROCESOS DEADLOCK EN EL GRAFO.

5.22 RECATE DE DEADLOCK

UNA VEZ QUE UN SISTEMA TIENE SENTADO UN DEADLOCK, ESTE SERA QUEBRANTADO PARA REMOVER UNA O MAS DE LAS CONDICIONES NECESARIAS. USUALMENTE VARIOS PROCESOS PIERDEN ALGUNOS O TODOS DE LOS TRABAJOS QUE ELLOS TIENEN CONSUMADOS. ESTE PUEDE SER UN PEQUENO PRECIO POR PAGAR, COMPARADO CON LA SALIDA DEL DEADLOCK EN PLAZA. EL RESCATE DEL DEADLOCK ES DIFICULTOSO POR UN NUMERO DE FACTORES:

- NO PUEDE SER DECLARADO DE QUE EL SISTEMA TENGA SENTADO UN DEADLOCK EN PRIMER LUGAR.
- MAS SISTEMAS TIENEN ESCASAS FACILIDADES PARA SUSPENDER UN PROCESO INDEFINIDO REMOVIDO DEL SISTEMA Y RESUMIENDOLO UN TIEMPO DESPUES. EN EFECTO VARIOS PROCESOS, TALES COMO LOS PROCESOS DE TIEMPO-REAL QUE PUEDEN FUNCIONAR CONTINUAMENTE SON SIMPLEMENTE NO RESPONSABLES DE SER SUSPENDIDOS O RESUMIDOS.
- UNIRSE SI EFECTIVAMENTE EXISTEN CAPACIDADES PARA SUSPENDER/RESUMIR, ESTARAN MAS CIERTAMENTE ENVUELTOS CONSIDERABLEMENTE ARRIBA Y REQUERIRA DE LA ATENCION DE UN ALTO OPERADOR ESPECIALIZADO. DESDE LUEGO NO SIEMPRE UN OPERADOR ESTA DISPONIBLE.
- EL RECOBRO DE UN DEADLOCK DE MODESTAS PROPORCIONES ENVOLVERA UN MONTO RAZONABLE DE TRABAJO; UN DEADLOCK A GRAN ESCALA CIENTOS DE PROCESOS TENSOS O UNIDOS REQUERIRAN UN MONTO DE TRABAJO.

EN SISTEMAS COMUNES, EL RESCATE ES ORDINARIAMENTE EJECUTADO POR FUERTES REMOVIMIENTOS A UN PROCESO DEL SISTEMA Y EL CUAL RECLAMA SUS 2 RECURSOS. EL PROCESO DE REMOVER ES ORDINARIAMENTE DESPERDICIADO, PERO LOS PROCESOS RESTANTES AHORA SERAN CAPACES DE COMPLETAR. ALGUNAS VECES ES NECESARIO REMOVER VARIOS PROCESOS HASTA QUE SUFICIENTES RECURSOS TIENEN QUE SER RECLAMADOS PARA PERMITIR EL RESTO DE LOS PROCESOS PARA TERMINAR. EL RESCATE, EN CIERTO MODO ES VISTO COMO UN LOS PROCESOS SERAN REMOVIDOS ACORDE A ALGUNA ORDEN DE PRIORIDADES. AQUI TAMBIEN MOSTRAREMOS ALGUNAS DIFICULTADES:

- LAS PRIORIDADES DE LOS PROCESOS DE DEADLOCK, PUEDEN NO EXISTIR ASI QUE EL OPERADOR PUEDE TOMAR UNA DESICION ARBITRARIA.
- LAS PRIORIDADES QUIZAS INCORRECTAS O ALGO TONTAS POR CONSIDERACIONES ESPECIFICAS TAL COMO DEADLINE SCHEDULING EN LA CUAL UN PROCESO RELATIVO DE BAJA PRIORIDAD TIENE UNA PRIORIDAD ALTA TEMPORALMENTE.
- ES OPTIMO QUE AL ESCOGER AL PROCESO PARA SER REMOVIDO REQUERIRA DE CONSIDERABLE ESFUERZO PARA SER DETERMINADO.

VEREMOS QUE LO MAS CONVENIENTE PARA UN RESCATE DE DEADLOCK SERA UN MECANISMO EFECTIVO DE SUSPENCION/RESUMEN. ESTO NOS PERMITIRA PONER TEMPORALMENTE HOLDS EN PROCESO Y ENTONCES ASI RESUMIR LOS PROCESOS TENIDOS FUERA DE LA PERDIDA DE TRABAJO PRODUCTIVO. UNA BUSQUEDA EN ESTA AREA ES IMPORTANTE POR OTRAS RAZONES QUE UN RESCATE DEL DEADLOCK. POR EJEMPLO. CONVIENE NECESARIAMENTE CREAR UN SISTEMA TEMPORALMENTE, EMPEZAR A FUNCIONAR EL SISTEMA OTRA VEZ DESDE EL PUNTO FUERA DE LA PERDIDA DE TRABAJO PRODUCTIVO. SUSPENDER/RESUMIR SERA DE USO AQUI. LA CARACTERISTICA DEL CHECK-POINT/RESTART ES APROVECHARSE EN VARIOS SISTEMAS QUE FACILITAN EL SUSPENDER/RESUMIR CON UNA PERDIDA DEL TRABAJO SOLAMENTE DESDE QUE EL ULTIMO CHECK-POINT ES TOMADO. PERO VARIOS SISTEMAS SON DISENADOS SIN TOMAR VENTAJA DE LA CAPACIDAD DE SUSPENDER /RESUMIR. NORMALMENTE REQUIERE CONCIENCIA DE ESFUERZO EN LA PARTE DE APLICACIONES DE SISTEMAS DESARROLLADOS PARA INCORPORAR AL CHECK-POINT/RESTART Y A MENOS QUE LOS TRABAJOS SEAN CORRIDOS REQUIEREN MUCHAS HORAS DE OPERACION.
DEADLOCK'S TENDRAN HORRENDAS CONSECUENCIAS EN CIERTOS SISTEMAS DE TIEMPO-REAL. UN PROCESO DE TIEMPO REAL CONTROLA EL SISTEMA MONITOREANDO UNA REFINERIA DE GASOLINA SIMPLE EL CUAL CONTINUARA PARA FUNCIONAR LA SEGURIDAD Y OPERACION PROPIA DE LA REFINERIA.

5.23 CONSIDERACIONES DEADLOCK EN FUTUROS SISTEMAS

EN RESIENTES SISTEMAS DE COMPUTADORAS, DEADLOCK ES GENERALMENTE VISTO COMO UNA MOLESTIA LIMITADA. MAS SISTEMAS IMPLEMENTAN EL DEADLOCK BASICO DE PREVENCION DE METODOS SUGERIDOS POR HAVENDER Y ESOS METODOS SON VISTOS SATISFACTORIAMENTE.
EN FUTUROS SISTEMAS, EL DEADLOCK SERA DISTANTE, UNA DE LAS MAS CRITICAS CONSIDERACIONES POR VARIAS RAZONES:
- FUTURAS COMPUTADORAS ESTAN ORIENTADAS HACIA OPERACIONES PARALELAS
ASINCRONICAS MAS QUE A LOS SISTEMAS DE SERIE DEL PASADO. EL MULTIPROCESAMIENTO SERA COMUN Y LA COMPUTACION EN PARALELO SERA PREVALENTE , HABRA MAS OPERACIONES EN CONCURRENCIAS.
- EN FUTUROS SISTEMAS LOS RECURSOS ALOJADOS TENDERAN A SER DINAMICOS. LOS PROCESOS SERAN CAPACES DE ADQUIRIR Y REALIZAR RECURSOS LIBREMENTE COMO SE NECESITEN. LOS USUARIOS NO TENDRAN QUE SABER MUCHO DE LOS RECURSOS QUE NECESITEN EN AVANCE DE SU EJECUCION. EN EFECTO CON LAS INTERFASES ANTICIPADAS USUARIO AMIGABLE EN EL FUTURO, MAS USUARIOS ESTARAN PREOCUPADOS CON LA CONSUMACION DE RECURSOS DE SUS PROCESOS.
- CON LA TENDENCIA EN AUMENTO DE SISTEMAS OPERATIVOS DISENADOS PARA VER DATOS COMO UN RECURSO DE LOS NUMEROSOS RECURSOS DE S.O. MAS MANEJADOS AUMENTARAN PRACTICAMENTE.

ADMINISTRACION DE ALMACENAMIENTO

6.1 INTRODUCCION

LA ORGANIZACION Y DIRECCION DE LA MEMORIA PRINCIPAL O MEMORIA PRIMARIA O MEMORIA REAL DE UN SISTEMA COMPUTACIONAL HA SIDO UNO DE LOS FACTORES MAS IMPORTANTES EN EL DISENO DE SISTEMAS OPERATIVOS.
LOS PROGRAMAS Y DATOS DEBEN ESTAR EN ALMACENAMIENTO PRINCIPAL EN ORDEN A SER EJECUTADOS O REFERENCIADOS DIRECTAMENTE. EL ALMACENAMIENTO SECUNDARIO ES COMUNMENTE UN DISCO, CINTA O TAMBOR Y PROPORCIONAN UNA CAPACIDAD MAS BARATA PARA LA ABUNDANCIA DE PROGRAMAS Y DATOS QUE DEBEN ESTAR DISPONIBLES PARA SU PROCESAMIENTO.

6.2 ORGANIZACION DE ALMACENAMIENTO

HISTORICAMENTE, EL ALMACENAMIENTO PRINCIPAL HA SIDO VISTO COMO UN RECURSO CARO. LO CUAL DEMANDA ATENCION DE LOS DISENADORES DE SISTEMAS, ESTE HA SIDO UN IMPULSO PARA EL MAXIMO USO DE ESTE COSTOSO RECURSO.
LOS PRIMEROS SISTEMAS FUERON ASOCIADOS CON MANEJO Y ORGANIZACION DE ALMACENAMIENTO PRINCIPAL PARA EL USO OPTIMO. PARA ORGANIZACION DE ALMACENAMIENTO NOSOTROS QUEREMOS LA MANERA EN QUE CADA ALMACEANAMIENTO PRINCIPAL ES VISTO.
SI NOSOTRTOS COLOCAMOS VARIOS USUARIOS EN EL ALMACENAMIENTO PRINCIPAL AL MISMO TIEMPO QUE SUCEDERIA ? SI VARIOS USUARIOS DE PROGRAMAS ESTAN EN ALMACENAMIENTO PRINCIPAL AL MISMO TIEMPO, NOSOTROS LE DARIAMOS A CADA UNO DE ELLOS LA MISMA CANTIDAD DE ESPACIO, O NOSOTROS DIVIDIRIAMOS EL ALMACENAMIENTO PRINCIPAL EN PORCIONES LLAMADAS PARTICIONES DE DIFERENTES TAMANOS ? SI NOSOTROS PARTICIONARAMOS EL ALMACENAMIENTO PRINCIPAL EN UNA MANERA RIGIDA CON PARTICIONES DEFINIDAS POR EXTENSOS PERIODOS DE TIEMPO O SI NOSOTROS SUMINISTRARAMOS A MAS PARTICIONES DINAMICAS PERMITIENDO AL SISTEMA COMPUTACIONAL ADOPTAR RAPIDAMENTE LOS CAMBIOS EN LAS NECESIDADES DE JOBS DE LOS USUARIOS. CUANDO NOSOTROS REQUERIMOS QUE LOS JOBS DE LOS USUARIOS SEAN DISENADOS A CORRER EN UNA PARTICION ESPECIFICA, ASI NOSOTROS PERMITIMOS A LOS JOBS CORRER EN CUALQUIER LUGAR QUE ELLOS QUEPAN. CUANDO NOSOTROS REQUERIMOS QUE CADA TRABAJO SEA COLOCADO EN UN BLOCK CONTIGUO DE UNA LOCALIZACION DE ALMACENAMIENTO, ASI NOSOTROS PERMITIMOS JOBS A SER SEPARADOS PARCIALMENTE EN BLOCKS SEPARADOS Y COLOCADOS EN CUALQUIER SLOT DISPONIBLE EN ALMACENAMIENTO PRINCIPAL.
LOS SISTEMAS HAN SIDO CONSTRUIDOS IMPLEMENTANDO CADA UNO DE ESTOS ESQUEMAS.

6.3 DIRECCIONAMIENTO DE ALMACENAMIENTO

SIN HACER CASO DE QUE EL ESQUEMA DE ALMACENAMIENTO ORGANIZACIONAL QUE ADAPTAMOS NOSOTROS PARA UN SISTEMA PARTICULAR, NOSOTROS DEBEMOS DECIDIR QUE ESTRATEGIAS USAR PARA OBTENER EL DESEMPENO OPTIMO. LAS ESTRATEGIAS DE DIRECCIONAMIENTO DE ALMACENAMIENTO DETERMINAN COMO UNA ORGANIZACION DE ALMACENAMIENTO PARTICULAR FUNCIONA BAJO VARIAS POLITICAS.
CUANDO NOSOTROS OBTENEMOS UN PROGRAMA NUEVO PARA COLOCARLO EN MEMORIA, SI NOSOTROS LO OBTENEMOS CUANDO EL SISTEMA ESPECIFICAMENTE PREGUNTA POR EL O SI NOSOTROS TENTATIVAMENTE NOS ANTICIPAMOS AL SISTEMA SOLICITADO. EN QUE PARTE DE ALMACENAMIENTO PRINCIPAL COLOCARIAMOS EL SIGUIENTE PROGRAMA A SER CORRIDO ? SI NOSOTROS COLOCAMOS PROGRAMAS LO MAS APRETADAMENTE POSIBLE DENTRO DE SLOTS DE MEMORIA DISPONIBLES PARA MINIMIZAR EL ESPACIO USADO O SI NOSOTROS COLOCAMOS PROGRAMAS RAPIDAMENTE COMO SEA POSIBLE PARA MINIMIZAR EL TIEMPO DE EJECUCION.
SI UN NUEVO PROGRAMA NECESITA SER PUESTO EN ALMACENAMIENTO PRINCIPAL Y SI EL ALMACENAMIENTO PRINCIPAL ESTA LLENO, CUAL DE LOS OTROS PROGRAMAS DESPLAZAREMOS ?. PODRIAMOS NOSOTROS REPONER EL PROGRAMA MAS VIEJO O PODRIAMOS REPONER AQUELLOS QUE SON MENOS FRECUENTEMENTE USADOS O AQUELLOS QUE SON MENOS RECIENTEMENTE USADOS ? LOS SISTEMAS HAN SIDO IMPLEMENTADOS USANDO CADA UNA DE ESTAS ESTRATEGIAS DE MANEJO DE ALMACENAMIENTO.

6.4 ALMACENAMIENTO JERARQUICO

EN LOS 50's Y 60's EL ALMACENAMIENTO PRINCIPAL ERA USUALMENTE EL CORAZON DE LA MEMORIA MAGNETICA, ERA MUY CARA. UNA DESICION DE QUE COMO EL ALMACENAMIENTO PRINCIPAL A COLOCARSE SOBRE UN SISTEMA COMPUTACIONAL FUE HECHO CUIDADOSAMENTE. UNA INSTALACION NO PUEDE COMPRAR MAS DE LO QUE PODRIA PROPORCIONAR, PERO SI TUBO QUE COMPRAR LO SUFICIENTE PARA SOPORTAR EL S.O. Y DARSELO A UN NUMERO DE USUARIOS.
LA META FUE COMPRAR LA MINIMA CANTIDAD DE SOPORTE ADECUADAMENTE DE USUARIOS CARGADOS ANTICIPADAMENTE DENTRO DEL CONTRASTE ECONOMICO DE LAS INSTALACIONES.
LOS PROGRAMAS Y DATOS NECESITAN ESTAR EN ALMACENAMIENTO PRINCIPAL EN ORDEN PARA SER EJECUTADOS O REFERENCIADOS. LOS PROGRAMAS O DATOS QUE NO NECESITEN INMEDIATAMENTE SER USDADOS PUEDEN SER GUARDADOS EN ALMACENAMIENTO SECUNDARIO Y DESPUES SER PASADOS DENTRO DE ALMACENAMIENTO PRINCIPAL PARA SU EJECUCION O REFERENCIA.
EL ALMACENAMIENTO SECUNDARIO COMO CINTAS O DISCOS SON GENERALMENTE MENOS COSTOSOS QUE EL ALMACENAMIENTO PRINCIPAL

Y TIENEN MUCHA MAYOR CAPACIDAD. EL ALMACENAMIENTO PRINCIPAL PUEDE SER ACCESADO MAS RAPIDO QUE EL ALMACENAMIENTO SECUNDARIO.
EN SISTEMAS CON VARIOS NIVELES DE ALMACENAMIENTO, UN GRAN LANZE SE HACE SOBRE PROGRAMAS Y DATOS LOS CUALES SON REGRESADOS Y ADELANTADOS SOBRE LOS NIVELES. ESTE LANZAMIENTO CONSUME RECURSOS COMO TIEMPO DE CPU QUE DE OTRA MANERA SE PODRIA USAR EN USOS PRODUCTIVOS.
EN LOS 60's SE LLEGO A ACLARAR QUE EL ALMACENAMIENTO JERARQUICO PODRIA SER ENTENDIDO POR MAS DE UN NIVEL CON MEJORAMIENTOS DRAMATICOS EN DESEMPENO Y UTILIZACION. ESTE NIVEL ADICIONAL, EL CACHE, ES UN ALMACENAMIENTO DE ALTA VELOCIDAD QUE ES MUCHO MAS RAPIDO QUE EL ALMACENAMIETO PRINCIPAL. EL ALMACENAMIENTO CACHE ES EXTREMADAMENTE CARO COMPARADO CON EL ALMACENAMIENTO PRINCIPAL Y POR LO TANTO UNICAMENTE SON USADOS PEQUENOS CACHES.
EL ALMACENAMIENTO CACHE IMPONE MAS DE UN NIVEL DE SHUTTLING EN EL SISTEMA. LOS PROGRAMAS EN ALMACENAMIENTO PRINCIPAL SON LANZADOS AL CACHE DE ALTA VELOCIDAD ANTES DE QUE SE EMPIEZE A EJECUTAR. EN EL CACHE, ELLOS PUEDEN SER EJECUTADOS MUCHO MAS RAPIDO QUE EN EL ALMACENAMIENTO PRINCIPAL. LA ESPERANZA DE LOS DISENADORES USANDO EL CONCEPTO CACHE ES QUE EL OVERHEAD ENVUELTO EN LANZAMIENTO DE PROGRAMAS ADELANTE Y ATRAS SERA MUCHO MAS PEQUENO QUE EL AUMENTO OBTENIDO EN EL DESEMPENO POR LA RAPIDA EJECUCION POSIBLE EN EL CACHE.

6.5 ESTRATEGIAS DE MANEJO DE ALMACENAMIENTO

LOS RECURSOS CAROS SON MANEJADOS INTENSIVAMENTE PARA ACTIVAR MEJOR USO. LA PRINCIPAL.
POR MUCHOS ANOS, EL JUICIO CONVENCIONAL A DEMANDADO EL FETCH EN EL CUAL LA SIGUIENTE PIEZA DEL PROGRAMA DE DATOS ES LLEVADA AL ALMACENAMIENTO PRINCIPAL CUANDO ESTE ES REFERENCIADO POR UN PROGRAMA CORRIENDO. ESTE SE EMPEZO A ACCESAR DESDE QUE NOSOTROS NO PODIAMOS GENERALMENTE PREDECIR HACIA DONDE IRIA EL SIGUIENTE CONTROL DEL PROGRAMA, EL OVERHEAD ENVUELTO, HACIENDO SUPOSICIONES Y ANTICIPANDO EL FUTURO PODRIAMOS EXCEDERNOS LEJOS LOS BENEFICIOS ESPERADOS.
AHORA MUCHAS INVESTIGACIONES PREDICEN QUE EL FETCH ANTICIPADO ARROJARA UNA MEJOR EJECUCION DEL SISTEMA.
LAS ESTRATEGIAS DE COLOCACION CONCERNIDAS DETERMINAN DONDE EL ALMACENAMIENTO PRINCIPAL HA DE COLOCAR UN PROGRAMA DE UTILIDAD. EN ESTOS PUNTOS NOSOTROS DEMOSTRAMOS LAS CONSIDERACIONES DE LAS ESTRATEGIAS DE COLOCACION DE ALMACENAMIENTO QUE SON DONDE PRIMERO SE ACOMODE (FIRST-FIT), DONDE MEJOR SE ACOMODE (BEST-FIT) Y DONDE PEOR SE ACOMODE (WORST-FIT).
LAS ESTRATEGIAS DE REPOSICION SON CONCERNIDAS DETERMINANDO CUAL PIEZA DEL PROGRAMA O DATOS SON DESPLAZADOS PARA HACER LUGAR A PROGRAMAS DE UTILIDAD.

6.6 LOCALIZACIONES DE ALMACENAMIENTO CONTIGUOS VS NO CONTIGUOS

LOS SISTEMAS COMPUTACIONALES REQUIEREN DE LOCALIZACIONES DE ALMACENAMIENTO CONTIGUOS DONDE CADA PROGRAMA TIENE QUE OCUPAR UN SIMPLE BLOCK CONTIGUO DE LOCALIZACIONES DE ALMACENAMIENTO, ESTO NO FUE HASTA QUE LA MULTIPROGRAMACION DE VARIABLES DE PARTICION ATENTARON QUE LAS LOCALIZACIONES DE ALMACENAMIENTO NO CONTIGUAS LLEGARIAN A SER DE UTILIDAD.
EN LAS LOCALIZACIONES DE ALMACENAMIENTO NO CONTIGUAS, UN PROGRAMA ES DIVIDIDO DENTRO DE VARIOS BLOCKS O SEGMENTOS QUE PUEDEN SER COLOCADOS DIRECTOS EN ALMACENAMIENTO PRINCIPAL EN LUGARES NO NECESARIAMENTE ADYACENTES UNOS DE OTROS. ESTO ES MAS DIFICIL PARA EL SISTEMA OPERATIVO PARA ENCONTRAR LOCALIZACIONES DE ALMACENAMIENTO NO CONTIGUOS. EL BENEFICIO ES QUE SI EL ALMACENAMIENTO PRINCIPAL TIENE MUCHOS PEQUEMOS HOYOS DISPONIBLES PREFERIBLEMENTE DE UN SIMPLE LARGO HOYO, ENTONCES LOS S.O. PUEDEN CARGAR Y EJECUTAR UN PROGRAMA QUE DE OTRO MODO TENDRIA NECESIDAD DE ESPERAR.

6.7 LOCALIDAD DE ALMACENAMIENTO CONTIGUO PARA UN SOLO USUARIO

LOS SISTEMAS COMPUTACIONALES PERMITEN SOLAMENTE A UNA PERSONA A UN TIEMPO USAR LA MAQUINA. TODOS LOS RECURSOS DE LA MAQUINA FUERON DISPUESTOS AL USUARIO. EL GIRO DE USO DE LAS COMPUTADORAS FUE MAS FRANCO PORQUE LOS USUARIOS TENIAN LA MAQUINA ENTERA, EL USUARIO CARGABA TODOS LOS RECURSOS USARA O NO TODOS ESTOS RECURSOS. LOS GIROS NORMALES DE LOS MECANISMOS FUERON BASADOS EN TIEMPO DE RELOJ DE PARED. LA MAQUINA ERA DADA AL USUARIO UN INTERVALO DE TIEMPO Y ERA CARGADO A UN RANGO DE HORARIO. ALGUNOS DE LOS SISTEMAS DE AHORA TIENEN UN MAS COMPLEJO GIRO DE ALGORITMOS.
ORIGINALMENTE CADA USUARIO ESCRIBE TODOS LOS CODIGOS NECESARIOS PARA IMPLEMENTAR UNA APLICACION PARTICULAR INCLUYENDO LAS INSTRUCCIONES I/O.
MUY RAPIDAMENTE, EL CODIGO I/O NECESITA LAS IMPLEMENTACIONES DE FUNCIONES BASICAS CONSOLIDADAS DENTRO DEL SISTEMA DE CONTROL I/O (IOCS).
LOS USUARIOS DESEARAN NO HACER EL CODIGO DE INSTRUCCIONES I/O MAS BAJO QUE LAS INSTRUCCIONES DIRECTAMENTE, ELLOS LLAMARON A LAS RUTINAS IOCS PARA HACER EL TRABAJO REAL, ESTO SIMPLIFICO Y AGILIZO EL CODIGO DE PROCESO. LAS IMPLEMENTACIONES DE LOS SISTEMAS DE CONTROL I/O PUDIERON HABER SIDO EL COMIENZO DEL CONCEPTO DE SISTEMAS OPERATIVOS.
LOS PROGRAMAS SON LIMITADOS EN TAMANOS A LA CANTIDAD DE ALMACENAMIENTO PRINCIPAL PERO ES POSIBLE CORRER PROGRAMAS GRANDES EN EL ALMACENAMIENTO PRINCIPAL USANDO OVERLAYS (EXTENDER LA MEMORIA). SI UNA SECCION DE UN PROGRAMA

PARTICULAR NO ES NECESITADO POR LA DURACION DE UN PROGRAMA EN EJECUCION, ENTONCES OTRA SECCION DEL PROGRAMA PUEDE SER ADQUIRIDA DESDE EL ALMACENAMIENTO SECUNDARIO A OCUPAR EL ALMACENAMIENTO USADO POR LA SECCION DEL PROGRAMA QUE NO ES NECESITADO.
LOS OVERLAYS DAN AL PROGRAMADOR UN CAMINO A EXTENDERSE LIMITADAMENTE EN ALMACENAMIENTO PRINCIPAL. PERO LOS OVERLAYS MANUALES REQUIEREN CUIDADOSAMENTE DE LA PLANEACION DE TIEMPO CONSUMIDO. UN PROGRAMA CON UNA ESTRUCTURA SOFISTICADA DE OVERLAY PUEDE SER DIFICIL DE MODIFICAR.

6.8 SISTEMAS DE PROTECCION EN USUARIOS SIMPLES

EN UN SISTEMA DE LOCALIZACIONES CONTIGUAS DE UN USUARIO SIMPLE, EL USUARIO TIENE EL CONTROL COMPLETO SOBRE TODO EL ALMACENAMIENTO PRINCIPAL. EL ALMACENAMIENTO ES DIVIDIDO DENTRO DE PORCIONES SOSTENIDAS POR LOS USUARIOS DE LOS PROGRAMAS Y POR PORCIONES NO USADAS. LA PREGUNTA SOBRE PROTECCION ES SIMPLE, COMO PODRIA SER PROTEGIDO EL S.O. DE LA DESTRUCCION POR LOS USUARIOS DE PROGRAMAS ? SI UN USUARIO VA POR MAL CAMINO, PUEDE DESTRUIR EL S.O. SI ESTO SUCEDE ES FATAL Y EL USUARIO NO PUEDE PROCEDER, ENTONCES EL USUARIO SABRA QUE ESTA EN UN ERROR SI TERMINADA LA EJECUCION ARREGLA EL PROBLEMA Y SE RETIRA DEL JOB. EN ESTAS CIRCUNSTANCIAS LA NECESIDAD PARA LA PROTECCION DEL S.O. ES IMPORTANTE.
PERO SUPONGAMOS QUE EL USUARIO DESTRUYE EL S.O. MAS "GENTILMENTE". POR EJEMPLO, SUPONGAMOS QUE CIERTAS RUTINAS I/O SON ACCIDENTALMENTE CAMBIADAS, ENTONCES TODOS LOS REGISTROS DE SALIDA (OUTPUT) PUEDEN SER TRUNCADOS, EL PUEDE SEGUIR CORRIENDO SI ESTOS RESULTADOS NO SON EXAMINADOS HASTA QUE SON COMPLETADOS, ENTONCES LOS RECURSOS DE LA MAQUINA PUEDEN SER DERROCHADOS. PEOR TODAVIA, EL PERJUICIO PARA EL S.O. PUEDE CAUSAR SALIDAS O SER PRODUCIDAS QUE NO PUEDEN FACILMENTE SER DETERMINADAS O SER INEXACTAS.
ESTO, SEGURAMENTE ES CLARO QUE EL S.O. DEBERA SER PROTEGIDO DE LOS USUARIOS. LA PROTECCION ES IMPLEMENTADA POR EL USO DE REGISTROS FRONTERA (BOUNDS REGISTER) DENTRO DEL CPU, A CADA TIEMPO UN USUARIO DE PROGRAMAS REFERENCIA UNA DIRECCION DE ALMACENAMIENTO, EL REGISTRO FRONTERA ES CHECADO A SER CIERTO QUE EL USUARIO NO ESTE CERCA DE DESTRUIR EL S.O., LOS REGISTROS FRONTERA CONTIENEN LA DIRECCION DE LA INSTRUCCION MAS ALTA USADA POR EL S.O. SI UN USUARIO TRATA DE ENTRAR AL S.O. ENTONCES LA INSTRUCCION ES INTERPRETADA Y EL JOB TERMINA CON UN MENSAJE DE ERROR.
POR SUPUESTO QUE EL USUARIO NECESITA ENTRAR AL S.O. TIEMPO A TIEMPO PARA OBTENER LOS SERVICIOS SEMEJANTES COMO I/O. ESTE PROBLEMA ES SOLVENTADO DANDO AL USUARIO INSTRUCCIONES ESPECIFICAS PARA CADA SERVICIO SOLICITADO DESDE EL S.O. EL

USUARIO QUERIENDO LEER DESDE LA CINTA MANDARA UNA INSTRUCCION PREGUNTANDO AL S.O. QUE HACER A FAVOR DEL USUARIO. EL S.O. EJECUTARA LA FUNCION DESEADA Y ENTONCES RETORNARA EL CONTROL AL USUARIO DEL PROGRAMA.
COMO LOS S.O. TIENDEN A SER MAS COMPLICADOS HA SIDO NECESARIO IMPLEMENTAR MAS MECANISMOS SOFISTICADOS PARA PROTEGER AL S.O. DE LOS USUARIOS Y LA PROTECCION DE USUARIOS UNOS DE OTROS. NOSOTROS DISCUTIREMOS ESTOS MECANISMOS CON MAS DETALLE MAS ADELANTE.

6.9 PROCESAMIENTO BATCH (LOTES) SIMPLE

LOS DATOS GENERALMENTE REQUIEREN CONSIDERABLE TIEMPO DE SETUP MIENTRAS EL S.O. ES CARGADO, CUANDO LOS JOBS TERMINAN SU EJECUCION ESTOS REQUIEREN DE CONSIDERABLE CANTIDAD DE (TEARDOWN TIME). LOS DISEÑADORES DE SISTEMAS REALIZARON LA TRANSICION DE JOB A JOB AUTOMATICA, ESTO REDUCIA EL TIEMPO CONSIDERABLEMENTE ENTRE JOB Y JOB USADO. LOS JOBS SON AGRUPADOS EN LOTES (BATCH) PARA LA EJECUCION POR EL S.O. UN PROCESADOR DE JOBS LEE LOS ESTATUTOS DE CONTROL DEL LENGUAJE Y FACILITA EL TIEMPO DE SETUP DEL PROXIMO JOB. EL SISTEMA DE PROCESAMIENTO EN BATCH ES ALTAMENTE USADO EN SISTEMAS COMPUTACIONALES Y HAN AYUDADO HA DEMOSTRAR EL VALOR REAL DE LOS SISTEMAS OPERATIVOS Y EL MANEJO DE LOS RECURSOS.

6.10 MULTIPROGRAMACION CON PARTICIONES FIJAS

CUALQUIER PROCESAMIENTO EN BATCH (LOTES), EN SISTEMAS DE UN SOLO USUARIO USA CONSIDERABLE CANTIDAD DE RECURSOS DE CPU HASTA QUE UN RECURSO DE I/O ES NECESITADO, CUANDO UN RECURSO DE I/O ES REQUERIDO A MENUDO EL JOB NO PUEDE CONTINUAR HASTA QUE HAYA REALIZADO LA LECTURA O ESCRITURA. LAS VELOCIDADES DE ENTRADA Y SALIDA SON EXTREMADAMENTE BAJAS CON LAS VELOCIDADES DEL CPU. LOS DISEÑADORES HAN VISTO QUE SE PODRIA UTILIZAR MAS EL CPU CON UNA ADMINISTRACION MAS INTENSIVA. ESTE TIEMPO OCIOSO SE EMPLEA PARA IMPLEMENTAR SISTEMAS DE MULTIPROGRAMACION EN LOS CUALES VARIOS USUARIOS SIMULTANEAMENTE TERMINAN EL USO DE RECURSOS DEL SISTEMA.
LOS JOB ESPERANDO UN RECURSO DE I/O PODRIAN CEDER TIEMPO DE CPU A OTRO JOB LISTO PARA HACER CALCULOS SI REALMENTE ESTA ESPERANDO.
ASI QUE AMBOS PROCESOS DE I/O Y CPU PUEDEN OCURRIR SIMULTANEAMENTE. ESTO INCREMENTA EN GRAN ESCALA LA UTILIZACION DEL CPU Y PODER DE COMPUTO.
PARA TOMAR UNA MAXIMA VENTAJA DE MULTIPROGRAMACION PARA VARIOS JOBS ES NECESARIO QUE LOS JOBS RECIDAN EN LA MEMORIA DE LA COMPUTADORA A UN TIEMPO. ASI QUE CUANDO UN JOB REQUIERE I/O, EL CPU PUEDE DETERMINAR INMEDIATAMENTE EL SWITCHEO PARA OTRO JOB Y HACER LOS CALCULOS SIN RETARDO.

CUANDO ESTE NUEVO JOB CEDE EL CPU OTRO, ESTA LISTO PARA USARLO.
LA MULTIPROGRAMACION A MENUDO REQUIERE DE MAS MEMORIA PRINCIPAL QUE LA DE UN SISTEMA DE UN SOLO USUARIO. SIN EMBARGO ESTA JUSTIFICADO POR LA MAYORIA DE LOS RECURSOS DE DISPOSITIVOS PERIFERICOS.

6.11 MULTIPROGRAMACION CON PARTICION FIJA TRASLACION Y CARGA ABSOLUTAS

EN LOS SISTEMAS DE MULTIPROGRAMACION INICIALES, LA MEMORIA PRINCIPAL ESTUVO DIVIDIDA EN UN NUMERO DE PARTICIONES DE TAMANO FIJO, CADA PARTICION PODIA RETENER UN JOB. EL CPU SWITCHEABA RAFIDAMENTE ENTRE LOS USUARIOS PARA CREAR LA ILUSION DE MULTIPROGRAMACION.
LOS JOBS SON TRASLADADOS CON COMPILADORES Y ENSAMBLADORES ABSOLUTOS PARA CORRER SOLAMENTE EN UNA PARTICION ESPECIFICA. SI UN JOB ESTA LISTO PARA CORRER Y UNA PARTICION ESTA OCUPADA ENTONCES EL JOB TIENE QUE ESPERAR AUN SI OTRA PARTICION ESTA DISPONIBLE, ESTE RESULTADO UTILIZA MUCHO RECURSO DE ALMACENAMIENTO.

6.12 MULTIPROGRAMACION CON PARTICIONES FIJAS TRASLACION RELOCALIZABLE Y CARGA

LOS COMPILADORES, ENSAMBLADORES Y CARGADORES RELOCALIZABLES SON USADOS PARA PRODUCIR PROGRAMAS RELOCALIZABLES QUE PUEDEN CORRER EN CUALQUIER PARTICION DISPONIBLE QUE ES LO SUFICIENTEMENTE GRANDE PARA RETENER EL PROGRAMA. UN EJEMPLO DE UTILIZACION DE MEMORIA POBRE EN MULTIPROGRAMACION CON PARTICION FIJA CON TRASLACION Y CARGA ABSOLUTA.

6.13 FRAGMENTACION EN MULTIPROGRAMACION CON PARTICION FIJA

LA FRAGMENTACION DE MEMORIA OCURRE EN CUALQUIER COMPUTADORA SIN HACER CASO DE LA ORGANIZACION DE LA MEMORIA. EN SISTEMAS DE MULTIPROGRAMACION CON PARTICIONES FIJAS, OCURRE LA FRAGMENTACION CUALQUIERA PORQUE EL USUARIO CON LOS JOBS NO LLENA COMPLETAMENTE LA PARTICION DESIGNADA O CUANDO UNA PARTICION SIN USO ES DEMASIADO PEQUENA PARA RETENER EL JOB ESPERANDO.

6.14 MULTIPROGRAMACION CON PARTICION VARIABLE

DISENADORES DE SISTEMAS OPERATIVOS HAN OBSERVADO EL PROBLEMA CON MULTIPROGRAMACION CON PARTICION FIJA, DECIDIERON QUE UNA MEJORA OBVIA PUEDE SER PERMITIR A LOS JOB OCUPAR TAL ESPACIO COMO ES NECESITADO. LOS LIMITES FIJOS NO PODRIAN SER VISTOS. ESTE ESQUEMA ES LLAMADO MULTIPROGRAMACION CON PARTICION VARIABLE.

6.15 UNIENDO HUECOS

CUANDO UN JOB TERMINA EN UN SISTEMA DE MULTIPROGRAMACION DE PARTICION VARIABLE, PODEMOS CHECAR EL ALMACENAMIENTO LIBRE (HUECOS). SI ES ASI ENTONCES PODEMOS REGISTRAR EN LA LISTA DE ALMACENAMIENTO LIBRE YA SEA (1) UN HUECO ADICIONAL O (2) UN HUECO SENCILLO REFLEJANDO LA FUSION DEL HUECO EXISTENTE Y EL NUEVO HUECO ADYACENTE.
EL PROCESO DE FUSIONAR HUECOS ADYACENTES PARA FORMAR UN SOLO HUECO MAS GRANDE ES LLAMADO UNION. AL UNIR HUECOS RECLAMAMOS BLOCKS CONTIGUOS DE ALMACENAMIENTO MAS GRANDES POSIBLES.

6.16 COMPACTACION DE ALMACENAMIENTO

ASI COMO LOS HUECOS SON UNIDOS, ES FRECUENTE EL CASO DE QUE LOS HUECOS SEPARADOS DISTRIBUIDOS POR TODO LA MEMORIA PRINCIPAL CONSTITUYE UNA CANTIDAD SIGNIFICANTE DE ALMACENAMIENTO. ALGUNAS VECES CUANDO UN JOB PIDE UNA CIERTA CANTIDAD DE MEMORIA PRINCIPAL NINGUN HUECO INDIVIDUAL ES LO SUFICIENTEMENTE GRANDE PARA CONTENER AL JOB, AUNQUE LA SUMA DE TODOS LOS HUECOS ES GRANDE QUE EL ALMACENAMIENTO NECESITADO POR EL NUEVO JOB. LA TECNICA DE COMPACTACION DE ALMACENAMIENTO ENVUELVE EL MOVER TODAS LAS AREAS DE ALMACENAMIENTO OCUPADAS A UN FINAL EXTREMO O AL OTRO DE MEMORIA PRINCIPAL. ESTO DEJA UN HUECO DE ALMACENAJE LIBRE SENCILLO Y GRANDE EN LUGAR DE LOS NUMEROSOS HUECOS PEQUENOS COMUNES EN LA MULTIPROGRAMACION DE PARTICION VARIABLE.
AHORA TODO EL ALMACENAMIENTO LIBRE DISPONIBLE ESTA CONTIGUO DE TAL MANERA QUE UN JOB QUE RESULTA DE LA COMPACTACION. ALGUNAS VECES, LA COMPACTACION DE ALMACENAMIENTO ES DRAMATICAMENTE DEFINIDA COMO "BURPING DEL ALMACENAMIENTO" MAS CONVENCIONALMENTE ESTO ES LLAMADO RECOLECCION DE BASURA.

DESVENTAJAS AL EFECTUAR LA COMPACTACION:

- CONSUME RECURSOS DEL SISTEMA QUE PODRIAN DE LO CONTRARIO SER USADOS PRODUCTIVAMENTE.
- EL SISTEMA DEBE PARAR POR COMPLETO MIENTRAS EJECUTA LA COMPACTACION. ESTO PUEDE RESULTAR EN TIEMPOS DE RESPUESTA ERATICOS PARA USARLOS INTERACTIVOS Y PODRIA SER DEVASTADOR EN SISTEMAS DE TIEMPO REAL.
- LA COMPACTACION ENVUELVE LA REUBICAION DE LOS JOBS QUE ESTAN EN ALMACENAMIENTO. ESTO SIGNIFICA QUE LA REUBICACION DE LA INFORMACION ORDINARIAMENTE PERDIDA CUANDO UN PROGRAMA ES CARGADO AHORA DEBE SER MANTENIDA EN FORMA ACCESIBLE.
- CON UNA NORMAL, MEZCLA DE JOBS CAMBIANDO RAPIDAMENTE, ES NECESARIO COMPACTAR FRECUENTEMENTE. LOS RECURSOS DEL SISTEMA CONSUMIDOS NO PUEDEN JUSTIFICAR LOS BENEFICIOS DE LA COMPACTACION.

6.17 ESTRATEGIAS DE COLOCACION DE ALMACENAMIENTO

LAS ESTRATEGIAS DE COLOCACION DE ALMACENAMIENTO SON USADAS PARA DETERMINAR DONDE COLOCAR EN EL ALMACENAMIENTO PRINCIPAL LOS PROGRAMAS Y DATOS QUE ESTAN LLEGANDO.
-ESTRATEGIA DONDE MEJOR SE ACOMODE (BEST-FIT)
UN JOB QUE ESTA INGRESANDO ES COLOCADO EN EL HUECO EN ALMACENMIENTO PRINCIPAL EN EL CUAL ESTE (EL JOB) SE ACOMODA MEJOR Y DEJA LA CANTIDAD MAS PEQUENA DE ESPACIO SIN USAR. PARA MUCHA GENTE, BEST-FIT PARECE SER INTUITIVAMENTE LA ESTRATEGIA MAS ATRAYENTE.
-ESTRATEGIA DONDE PRIMERO SE ACOMDE (FIRST-FIT)
UN JOB QUE ESTA INGRESANDO ES COLOCADO EN EL ALMACENAMIENTO PRINCIPAL EN EL PRIMER HUECO DISPONIBLE LO SUFICIENTEMENTE GRANDE PARA CONTENERLO, FIRST-FIT TAMBIEN TIENE CIERTA ATRACCION INTUITIVA YA QUE PERMITE QUE LA DESICION DE COLOCACION SEA HECHA RAPIDAMENTE.
-ESTRATEGIA DONDE PEOR SE ACOMODE (WORST-FIT)
AL PRINCIPIO ESTA PARECE SER UNA ELECCION CAPRICHOSA. DESPUES DE UN EXAMEN MAS CERCANO, WORDS-FIT DICE QUE SE COLOQUE UN PROGRAMA EN ALMACENAMIENTO PRINCIPAL EN EL HUECO EN EL CUAL SE AJUSTE PESIMAMENTE, ESTO ES EL HUECO MAS GRANDE POSIBLE. LA ATRACCION INTUITIVA ES SIMPLE; DESPUES DE COLOCAR EL PROGRAMA EN ESTE GRAN HUECO, EL HUECO RESTANTE TAMBIEN ES FRECUENTEMENTE GRANDE Y ASI PUEDE CONTENER UN NUEVO PROGRAMA RELATIVAMENTE GRANDE.

6.18 MULTIPROGRAMACION CON ALMACENAMIENTO (STORAGE) SWAPPING

EN CADA UNO DE LOS ESQUEMAS DE MULTIPROGRAMACION DISCUTIDA HASTA AHORA, LOS PROGRAMAS DE USUARIO PERMANECEN EN MEMORIA PRINCIPAL HASTA SU TERMINACION, OTRO ESQUEMA LLAMADO "SWAPPING" NO TIENE ESTE REQUERIMIENTO.
EN ALGUNOS SISTEMAS DE SWAPPING UN JOB OCUPA EL ALMACENAMIENTO PRINCIPAL INMEDIATAMENTE, ESE JOB CORRE HASTA QUE NO PUEDE CONTINUAR MUCHO MAS TIEMPO Y LUEGO ABANDONA AMBOS, EL ALMACENAMIENTO Y EL CPU AL SIGUIENTE JOB.
ASI EL ALMACENAMIENTO COMPLETO ES DEDICADO A UN JOB POR UN BREVE PENADO, ESE JOB ES LUEGO REMOVIDO (ESTO ES SWAPPED OUT) Y EL SIGUIENTE JOB ES TRAIDO DENTRO (ESTO ES SWAPPED IN). UN JOB SERA NORMALMENTE CAMBIADO DENTRO Y FUERA MUCHAS VECES ANTES DE QUE ESTE COMPLETO. MUCHOS DE LOS PRIMEROS SISTEMAS DE TIEMPO COMPARTIDO FUERON IMPLEMENTADOS CON ESTA TECNICA DE SWAPPING, FUE POSIBLE GARANTIZAR TIEMPOS DE RESPUESTA RAZONABLES PARA RELATIVAMENTE POCOS USUARIOS, PERO LOS DISENADORES SABIAN QUE SERIAN NECESITADAS MEJORES TECNICAS PARA CONTENER GRANDES NUMEROS DE USUARIOS. LOS SISTEMAS SWAPPING DE PRINCIPIOS DE LOS 60's CONDUJERON A LOS SISTEMAS DE PAGINACION, HOY DE USO COMUN. SISTEMAS SWAPPING

MAS SOFISTICADOS HAN SIDO DESARROLLADOS Y PERMITEN A VARIAS IMAGENES DE USUARIO PERMANECER EN ALMACENAMIENTO PRINCIPAL INMEDIATAMENTE. EN ESTOS SISTEMAS SU ALMACENAMIENTO ES NECESITADO PARA UNA IMAGEN DE USUARIO QUE SERA INGRESADO. CON UNA SUFICIENTE CANTIDAD DE ALMACENAMIENTO, LOS SISTEMAS REDUCEN GRANDEMENTE EL TIEMPO GASTADO DE SWAPPING.

ORGANIZACION DE ALMACENAMIENTO VIRTUAL

7.1 INTRODUCCION

EL TERMINO DE ALMACENAMIENTO VIRTUAL ES REALMENTE ASOCIADO CON LA HABILIDAD PARA DIRECCIONAR UN ESPACIO DE ALMACENAMIENTO MAYOR QUE EL DISPONIBLE EN ALMACENAMIENTO PRIMARIO DE UN PARTICULAR SISTEMA COMPUTACIONAL. EL ALMACENAMIENTO VIRTUAL NO ES UN CONCEPTO NUEVO, ESTE CONCEPTO EXISTIO POR PRIMERA VEZ EN EL SISTEMA DE COMPUTADORA ATLAS CONSTRUIDO EN LA UNIVERSIDAD DE MANCHESTER EN INGLATERRA EN 1960.
LOS 2 METODOS MAS COMUNES PARA LA IMPLEMENTACION DE ALMACENAMIENTO VIRTUAL ES SEGMENTACION Y PAGINACION, LOS CUALES SERAN DISCUTIDOS A DETALLE EN LAS SIGUIENTES SECCIONES.
ALGUNOS SISTEMAS DE ALMACENAMIENTO VIRTUAL USAN ALGUNAS DE ESTAS TECNICAS Y OTROS SISTEMAS USAN AMBAS. TODOS LOS SISTEMAS DE ALMACENAMIENTO VIRTUAL TIENEN EL ATRIBUTO DE QUE LAS DIRECCIONES DESARROLLADAS DURANTE LA EJECUCION DEL PROGRAMA NO NECESARIAMENTE SON LAS DIRECCIONES DISPONIBLES EN ALMACENAMIENTO PRIMARIO.
DE HECHO LAS DIRECCIONES VIRTUALES SON NORMALMENTE SELECCIONADAS DESDE UN CONJUNTO MAYOR DE DIRECCIONES QUE LAS DISPONIBLES EN ALMACENAMIENTO PRINCIPAL.

7.2 CONCEPTOS BASICOS DE ALMACENAMIENTO VIRTUAL

EL CONCEPTO DE LLAVE EN ALMACENAMIENTO VIRTUAL NO ESTA ASOCIADO A LAS DIRECCIONES REFERENCIADAS POR EL PROCESO DE EJECUCION DESDE EL ALMACENAMIENTO DISPONIBLE EN ALMACENAMIENTO PRIMARIO.
LAS DIRECCIONES REFERENCIADAS POR UN PROCESO DE EJECUCION SON LLAMADAS DIRECCIONES VIRTUALES. LAS DIRECCIONES DISPONIBLES EN ALMACENAMIENTO PRINCIPAL SON LLAMADAS DIRECCIONES REALES. EL RANGO DE DIRECCIONES VIRTUALES REFERENCIADAS POR UN PROCESO EN EJECUCION ES LLAMADO ESPACIO DE DIRECCIONES VIRTUALES DEL PROCESO. EL RANGO DE DIRECCIONES DISPONIBLES EN UN SISTEMA PARTICULAR DE COMPUTADORA ES LLAMADO ESPACIO DE DIRECCIONES REALES DE LA COMPUTADORA. AUNQUE LOS PROCESOS HACEN SOLO REFERENCIAS A DIRECCIONES VIRTUALES ELLOS DEBEN ACTUALMENTE EJECUTARSE EN ALMACENAMIENTO REAL. DE ESTE MODO LAS DIRECCIONES VIRTUALES DEBEN SER MAPEADAS DENTRO DE DIRECIONES REALES COMO PROCESOS EN EJECUCION. ESTO DEBE HACERSE RAPIDAMENTE DE LO CONTRARIO LA EJECUCION DEL SISTEMA DE COMPUTADORA SE DEGRADARIA EN NIVELES INTOLERABLES. DE ESTE MODO SE ELIMINARIAN MUCHAS

GANANCIAS DESDE EL USO DEL CONCEPTO DE ALMACENAMIENTO VIRTUAL EN PRIMER LUGAR.
VARIAS PROPUESTAS HAN SIDO DESARROLLADAS PARA LA ASOCIACION DE DIRECCIONAMIENTO VIRTUAL CON DIRECCIONAMIENTO REAL. TRASLACION DE DIRECCIONAMIENTO DINAMICO.
ESTO ES UN MECANISMO PARA CONVERTIR DIRECCIONES VIRTUALES A DIRECCIONES REALES COMO EN PROCESOS EN EJECUCION. TODOS ESTOS SISTEMAS TIENEN LA PROPIEDAD DE QUE LAS DIRECCIONES CONTIGUAS EN EL ESPACIO DE DIRECCIONAMIENTO VIRTUAL DEL PROCESO NO NECESITAN SER CONTIGUAS EN ALMACENAMIENTO REAL, ESTO ES LLAMADO CONTIGUEDAD ARTIFICIAL. DE ESTE MODO LOS USUARIOS NO TIENEN QUE PREOCUPARSE DE DONDE FUERON POSECIONADOS LOS PROCEDIMIENTOS Y LOS DATOS EN ALMACENAMIENTO REAL. EL USUARIO PUEDE ESCRIBIR SUS PROGRAMAS DE LA MANERA MAS NORMAL CONSIDERANDO LOS DETALLES DEL DISEÑO DE ALGORITMOS Y ESTRUCTURA DEL PROGRAMA, PERO IGNORANDO LOS DETALLES DE LA ESTRUCTURA FUNDAMENTAL DEL HARDWARE. LA COMPUTADORA ES O PUEDE SER VISTA EN UN SENTIDO LOGICO COMO UN IMPLEMENTADOR DE ALGORITMOS EN VEZ DEL SENTIDO FISICO COMO UN DISPOSITIVO CON CARACTERISTICAS UNICAS, ALGUNAS DE LAS CUALES PUEDEN IMPEDIR EL DESARROLLO DEL PROCESO DEL PROGRAMA.

¿como trabaja el?

7.3 ORGANIZACION DE ALMACENAMIENTO MULTINIVEL?

SI NOSOTROS PERMITIMOS QUE EL ESPACIO DE DIRECCIONAMIENTO VIRTUAL DE USUARIO SEA MAYOR QUE EL ESPACIO DE DIRECCIONAMIENTO REAL Y CIERTAMENTE NOSOTROS MANEJAMOS UN SISTEMA DE MULTIPROGRAMACION EFECTIVAMENTE CON MULTIPLES USUARIOS COMPARTIENDO LA FUENTE DE ALMACENAMIENTO REAL, ENTONCES NOSOTROS DEBEMOS PROPORCIONAR UN MEDIO PARA TENER PROGRAMAS Y DATOS EN UN GRAN ALMACENAMIENTO AUXILIAR.
ESTO NORMALMENTE SE REALIZA USANDO UN ESQUEMA DE ALMACENAMIENTO DE 2 NIVELES, EL PRIMER NIVEL ES UN ALMACENAMIENTO REAL EN EL CUAL LOS PROCESOS SE EJECUTAN Y SER REFERENCIADOS POR UN PROCESO EN EJECUCION. EL SEGUNDO CONSISTE DE UNA GRAN CANTIDAD DE MEDIOS DE ALMACENAMIENTO TAL COMO DISCOS, TAMBORES CAPACES DE SOSTENER PROGRAMAS Y DATOS QUE PUEDAN SER COLOCADOS TOTALMENTE EN UN ALMACENAMIENTO REAL FINITO A LA VEZ. ESTE NIVEL GENERALMENTE ES LLAMADO ALMACENAMIENTO AUXILIAR SECUNDARIO O ALMACENAMIENTO DE RESPALDO. CUANDO UN PROCESO ES EJECUTADO, EL CODIGO Y LOS DATOS SON ALMACENADOS EN ALMACENAMIENTO PRINCIPAL. DADO QUE EL ALMACENAMIENTO REAL ES COMPARTIDO POR MUCHOS PROCESOS Y QUE CADA PROCESO PUEDE TENER ESPACIO DE DIRECCIONAMIENTO VIRTUAL MUCHO MAS GRANDE QUE EL ALMACENAMIENTO REAL, ENTONCES SOLO UNA PEQUEÑA PORCION DE CADA PROGRAMA DE PROCESO Y DATOS PUEDEN SER MANTENIDOS EN EL ALMACENAMIENTO REAL EN UN TIEMPO.

7.4 MAPEO POR BLOCK

LOS MECANISMOS DE TRANSFORMACION DE DIRECCIONES ES DINAMICA, SE DEBEN DE MANTENER EN MAPAS ILUSTRANDO CUALES LOCALIZACIONES DE ALMACENAMIENTO VIRTUAL ESTAN ACTUALMENTE EN ALMACENAMIENTO REAL Y DONDE ESTAN. SI ESTE MAPEO FUE EJECUTADO BAJO LA BASE DE LA PALABRA POR PALABRA Y BYTE POR BYTE, ENTONCES LA INFORMACION DEL MAPEO SERIA MAS VOLUMINOSO TAL QUE ESTO REQUERIRIA MUCHO O MAS ALMACENAMIENTO REAL QUE LOS PROCESOS QUE ELLOS MISMOS REQUIEREN. UN METODO ES NECESITADO PARA REDUCIR LA CAPACIDAD DE INFORMACION DE MAPEO EN EL ORDEN EN QUE SE HIZO LA IMPLEMENTACION DE ALMACENAMIENTO VIRTUAL (WORTHWHILE).
NOSOTROS NO PODEMOS PERMITIR MAPEAR ELEMENTOS INDIVIDUALMENTE POR ESTO, LA INFORMACION ES AGRUPADA DENTRO DE BLOCKS Y EL SISTEMA MANTIENE EL TRACK DE DONDE EL ALMACENAMIENTO REAL TOMA VARIOS BLOCK DE ALMACENAMIENTO VIRTUAL. EL TAMANO DEL BLOCK MAS GRANDE ES LA FUNCION MAS PEQUENA DE ALMACENAMIENTO REAL QUE DEBE SER DESTINADA AL ALMACENAMIENTO DE LA INFORMACION MAPEADA.
HACIENDO BLOCKS MAS LARGOS SERIA MAS BAJO EL OVERHEAD DE ALMACENAMIENTO DEL MECANISMO DE MAPEO. PERO BLOCKS MAS GRANDES TOMARIAN MAS TIEMPO DE TRANSFERENCIA ENTRE EL ALMACENAMIENTO PRIMARIO Y EL SECUNDARIO Y CONSUMIR MAS ALMACENAMIENTO REAL POSIBLEMENTE LIMITANDO EL NUMERO DE PROCESOS QUE PUEDA COMPARTIR EL ALMACENAMIENTO REAL.
HAY ALGUNAS PREGUNTAS DE QUE SI LOS BLOCKS SERIAN DEL MISMO TAMANO O DE DIFERENTES TAMANOS CUANDO LOS BLOCKS SON DEL MISMO TAMANO ELLOS SON LLAMADOS PAGINAS Y LA ASOCIACION CON LA ORGANIZACION DE ALMACENAMIENTO VIRTUAL ES LLAMADO PAGINACION. CUANDO LOS BLOCKS SON DE DIFERENTES TAMANOS SON LLAMADOS SEGMENTOS Y LA ASOCIACION CON LA ORGANIZACION DE ALMACENAMIENTO VIRTUAL ES LLAMADA SEGMENTACION.
ALGUNOS SISTEMAS COMBINAN LAS 2 TECNICAS DE IMPLEMENTACION DE SEGMENTOS COMO ENTIDADES DE TAMANO VARIABLE COMPUESTAS DE PAGINAS DE TAMANO FIJO.
LAS DIRECCIONES EN UN SISTEMA DE MAPEO DE BLOCKS SON DE 2 DIMENCIONES. PARA REFERIRSE A UN PARTICULAR ELEMENTO, UN PROGRAMA ESPECIFICA EL BLOCK EN EL CUAL EL ELEMENTO RESIDE Y EL DESPLAZAMIENTO DEL ELEMENTO DESDE EL INICIO DEL BLOCK. UNA DIRECCION VIRTUAL v, ES DENOTADA POR EL PAR ORDENADO (b,d) DONDE b ES EL NUMERO DEL BLOCK EN EL CUAL EL ELEMENTO REFERENCIADO RECIDE Y d ES EL DESPLAZAMIENTO DESDE EL INICIO DEL BLOCK. LA TRANSICION DESDE UN DIRECCIONAMIENTO DE ALMACENAMIENTO VIRTUAL v=(b,d) A UNA DIRECCION DE ALMACENAMIENTO REAL R, PROCEDE. CADA PROCESO TIENE SU PROPIA TABLA DE MAPEO DE BLOCK MANTENIDA POR EL SISTEMA EN EL ALMACENAMIENTO REAL. UN REGISTRO ESPECIAL EN LA UNIDAD DE PROCESAMIENTO ES LLAMADO ORIGEN DE TABLA DE BLOCKS, EL CUAL ES CARGADO CON LA DIRECCION REAL DE LA TABLA DE MAPEO DE

BLOCK. LA TABLA DE MAPEO DE BLOCK CONTIENE UNA ENTRADA PARA CADA BLOCK DEL PROCESO Y SON MANTENIDAS EN ORDEN SECUENCIAL PARA EL BLOCK 0, BLOCK 1. AHORA EL NUMERO DE BLOCK, b ES AGREGADO A LA DIRECCION BASE, LA TABLA DE BLOCK DESDE LA DIRECCION REAL DE LA ENTRADA A LA TABLA DE MAPEO DE BLOCK PARA EL BLOCK b. ESTA ENTRADA CON LA DIRECCION REAL b PARA EL BLOCK b, EL DESPLAZAMIENTO DE AGREGADO AL PRINCIPIO DEL BLOCK EN LA DIRECCION b DESDE LA DIRECCION DESEADA, r=b'+d. ES IMPORTANTE NOTAR QUE EL MAPEO DE BLOCK ES EJECUTADO DINAMICAMENTE COMO LA EJECUCION DE UN PROCESO. SI NO SE HIZO UNA IMPLEMENTACION EFICIENTE, EL OVERHEAD PODRIA CAUSAR UNA DEGRADACION A LA EJECUCION TAL QUE ELIMINARIA LOS BENEFICIOS OBTENIDOS POR EL USO DE ALMACENAMIENTO VIRTUAL.

7.5 CONCEPTO BASICO DE PAGINACION

RECORDANDO LA COMPLEJIDAD DEL MANEJO DE BLOCKS DE TAMANO VARIABLE BAJO LA MULTIPROGRAMACION DE PARTICIONES VARIABLES NOS PERMITIMOS EMPEZAR CONSIDERANDO MAPEO DE BLOCKS DE TAMANO FIJO, PAGINACION. EN ESTA SECCION NOSOTROS CONSIDERAMOS LA PAGINACION PURA DISTINGIENDOLA DE LA PAGINACION EN UN SISTEMA CON SEGMENTACION Y PAGINACION.
UN DIRECCIONAMIENTO VIRTUAL EN UN SISTEMA DE PAGINACION ESTA EN UN PAR ORDENADO (p,d) DONDE p ES EL NUMERO DE PAGINA EN ALMACENAMIENTO VIRTUAL EN EL CUAL LOS ELEMENTOS REFERENCIADOS RESIDEN Y d ES EL DESPLAZAMIENTO CON PAGINA p EN EL CUAL EL ELEMENTO REFERENCIADO ES LOCALIZADO, UN PROCESO SE PUEDE EJECUTAR SI LA PAGINA ACTUAL ESTA EN ALMACENAMIENTO PRIMARIO. LAS PAGINAS SON TRANSFERIDAS DESDE EL ALMACENAMIENTO SECUNDARIO AL ALMACENAMIENTO PRIMARIO Y SON COLOCADAS EN BLOCKS EN ALMACENAMIENTO PRIMARIO LLAMADO MARCOS DE PAGINA (PAGE FRAMES), LAS PAGE FRAME EMPIEZAN EN UNA DIRECCION DE ALMACENAMIENTO REAL QUE ESTAN INTEGRADAS POR MULTIPLES PAGINAS DE TAMANO FIJO UNA PAGINA INCOMING PUEDE SER COLOCADA EN UNA PAGINA FRAME DISPONIBLE. LA TRASLACION DE DIRECCIONAMIENTO DINAMICO BAJO PAGINACION PROCEDE COMO SIGUE. UN PROCESO EN EJECUCION REFERENCIA DIRECCIONES DE ALMACENAMIENTO VIRTUAL v=(p,d), ESTA EN LA TABLA DE MAPEO DE PAGINA Y DETERMINA QUE LA PAGINA p ES UNA PAGINA FRAME p'.LA DIRECCION DE ALMACENAMIENTO REAL ES ENTONCES FORMADA POR LA CONCATENACION DE p Y d. AHORA CONSIDEREMOS ESTE PROCESO MAS A DETALLE EN PARTICULAR. COMUNMENTE NO PUEDEN ESTAR MUCHAS PAGINAS DE PROCESO EN ALMACENAMIENTO PRIMARIO A LA VEZ, LA TABLA DE MAPEO DE PAGINA DEBE INDICAR SI LA PAGINA REFERENCIADA ESTA O NO EN ALMACENAMIENTO PRIMARIO, DEBE INDICAR DONDE ESTA LOCALIZADA DE LO CONTRARIO DEBE INDICAR DONDE SE LOCALIZA EN ALMACENAMIENTO SECUNARIO. EL BIT DE REFERENCIA DE LA PAGINA r ES COLOCADO EN 0 SI LA PAGINA NO ESTA EN ALMACENAMIENTO

PRIMARIO Y ES PUESTO EN 1 SI LA PAGINA ESTA EN ALMACENAMIENTO PRIMARIO. SI LA PAGINA ESTA EN ALMACENAMIENTO PRIMARIO ENTONCES p' ES EL NUMERO DE PAGINA FRAME. NOTE QUE p'NO ES UNA ACTUAL DIRECCION EN ALMACENAMIENTO PRIMARIO. LA DIRECCION DE ALMACENAMIENTO PRIMARIO a, EN LA CUAL INICIA LA PAGINA FRAME, p. (ASUMIENDO EL TAMANO DE LA PAGINA COMO p) ES DADO COMO a=(p)(p') ASUMIENDO QUE LAS PAGINAS FRAME SON NUMERADAS COMO (0,1,2,3,..,ETC). SI EL ALMACENAMIENTO PRIMARIO ES MUY ESCASO, LA DIRECCION DE ALMACENAMIENTO SECUNDARIO DE LA PAGINA PUEDEN SER OMITIDOS DESDE LA TABLA DE MAPEO DE PAGINA.

7.6 TRASFORMACION DE DIRECCIONES DE PAGINACION POR MAPEO DIRECTO

EN ESTA Y EN LAS PROXIMAS SECCIONES, NOSOTROS CONSIDERAREMOS ALGUNAS TECNICAS PARA LA EJECUCION DE MAPEO DE PAGINA. PRIMERO SE CONSIDERARA EL MAPEO DIRECTO.
UN PROCESO EN EJECUCION REFERENCIA DIRECCIONES VIRTUALES v=(p,d) ANTES QUE EL PROCESO INICIE SU EJECUCION, EL SISTEMA OPERATIVO CARGA LAS DIRECCIONES DE ALMACENAMIENTO PRIMARIO DE LA TABLA DE MAPEO DE PAGINA. LA DIRECCION BASE, b DE LA TABLA DE MAPEO DE PAGINA ES FORMADA AL NUMERO DE PAGINA p PARA FORMAR LA DIRECCION EN ALMACENAMIENTO PRIMARIO, btp AL ENTRAR EN LA TABLA DE MAPEO PAGINA PARA LA PAGINA p, ESTA ENTRADA INDICA QUE EL MARCO DE PAGINA (PAGINA FRAME) p, ES CONCATENADA CON EL DESPLAZAMIENTO DE d, PARA FORMAR LA DIRECCION REAL r. SE DICE QUE ESTE ES UN EJEMPLO DE MAPEO DIRECTO DADO QUE LA TABLA DE MAPEO DE PAGINA CONTIENE ENTRADA PARA VARIAS PAGINAS EN EL ALMACENAMIENTO VIRTUAL DEL PROCESO.
LAS DIRECCIONES VIRTUALES SON TRANSFORMADAS Y LA DIRECCION BASE DE LA TABLA DE MAPEO DE PAGINA SON MANTENIDAS EN REGISTRO DE VELOCIDAD EN EL PROCESO DE CONTROL. POR LO TANTO EN LAS OPERACIONES QUE ENVUELVEN A ELLAS PUEDEN SER EJECUTADAS RAPIDAMENTE CON UN CICLO DE EJECUCION DE INSTRUCCION SIMPLE. PERO EL MAPEO DIRECTO DE LA TABLA DE MAPEO DE PAGINA EL CUAL PUEDE SER UN POCO LARGO, ORDINARIAMENTE ES COLOCADO EN EL ALMACENAMIENTO PRIMARIO. CONSECUENTEMENTE LA REFERENCIA A LA TABLA DE MAPEO DE LA PAGINA REQUIERE UN CICLO DE ALMACENAMIENTO PRIMARIO COMPLETO. DADO QUE EL TIEMPO DEL CICLO DE ALMACENAMIENTO PRIMARIO ORDINARIAMENTE REPRESENTA LA PARTE MAS LARGA DE UN CICLO DE EJECUCION DE UNA INSTRUCCION YA QUE SE REQUIERE OTRO CICLO DE EJECUCION DE ALMACENAMIENTO PRIMARIO PARA LA PAGINA DE MAPEO ENTONCES EL USO DE MAPEO DIRECTO DE LA TRANSFORMACION DE DIRECCION DE LA PAGINA PUEDE CAUSAR QUE EL SISTEMA DE COMPUTADORA EJECUTE PROGRAMAS MAS O MENOS A LA MITAD DE SU VELOCIDAD, ESTO ES INTOLERABLE.

POR LO TANTO, LOS METODOS DE TRANSFORMACION MAS RAPIDOS DEBEN SER USADOS. NO ES COMPLETAMENTE INVALIDO EL USO DE MAPEO DIRECTO DADO QUE SISTEMAS IMPLEMENTAN UN MAPEO DIRECTO COMPLETO EN LA TABLA DE PAGINA A UNA VELOCIDAD MUY ALTA EN ALMACENAMIENTO CACHE.

7.7 TRASLACION DE DIRECCIONES DE PAGINACION POR UN MAPEO ASOCIATIVO

UNA MANERA DE AUMENTAR LA VELOCIDAD DE TRASLACION DE DIRECCION DINAMICA ES COLOCAR LA ENTRADA DE LA TABLA DE MAPEO DE PAGINA DENTRO DE UNA ALMACENAMIENTO ASOCIATIVO QUE PUEDE TENER UN CICLO DE TIEMPO EN UN ORDEN DE MAGNITUD MAS RAPIDO QUE EL ALMACENAMIENTO PRIMARIO.
UN PROGRAMA DE EJECUCION REFERENCIA DIRECCIONES VIRTUALES v=(p,d). CADA ENTRADA EN ALMACENAMIENTO ASOCIATIVO ES SIMULTANEAMENTE BUSCADA POR PAGINA p ESTO RETORNA UNA p', COMO EL MARCO DE PAGINA (PAGE FRAME) QUE CORRESPONDE A LA PAGINA p,p', ES CONCATENADA CON d FORMANDO LA DIRECCION REAL, r. NOTE QUE LAS FLECHAS DENTRO DEL MAPEO ASOCIATIVO ACTUALMENTE ESTAN EN CADA CELDA DEL MAPA. ESTO INDICA QUE CADA CELDA DE ALMACENAMIENTO ASOCIATIVO ES SIMULTANEAMENTE PARA UNA PAREJA SOBRE p. ESTO CREA UN CARO ALMACENAMIENTO ASOCIATIVO.
OTRA VEZ, EL DILEMA ES QUE CADA TRANSFORMACION DE DIRECCION DINAMICA SE DEBE REALIZAR RAPIDAMENTE PARA HACER QUE EL CONCEPTO DE ALMACENAMIENTO VIRTUAL SEA FACTIBLE. EL USO DEL ALMACENAMIENTO CACHE PARA IMPLEMENTAR UN MAPEO DIRECTO PURO O EL USO DE UN ALMACENAMIENTO SUCEPTIBLE PARA IMPLEMENTAR UN MAPEO ASOCIATIVO PURO SON TAMBIEN COSTOSOS.
NOSOTROS NECESITAMOS UN ESQUEMA COMPLETO QUE OFREZCA LAS MEJORES VENTAJAS DE APROXIMACION AL ALMACENAMIENTO ASOCIATIVO O CACHE A UN BAJO COSTO.

7.8 TRASFORMACION DE DIRECCION DE PAGINACION CON COMBINACION DE MAPEO ASOCIATIVO Y DIRECTO

MUCHAS DE LAS DISCUSIONES A ESTE PUNTO HAN INTERVENIDO CON EL HARDWARE DE LA COMPUTADORA EL CUAL IMPLEMENTA EL ALMACENAMIENTO VIRTUAL EFICIENTEMENTE. LA VISTA DEL HARDWARE PRESENTADA HA SIDO MAS LOGICA QUE FISICA. NOSOTROS TOMAMOS EN CUENTA LA ESTRUCTURA PRECISA DE LOS DISPOSITIVOS, PERO SI TOMAMOS EN CUENTA LA ORGANIZACION FUNCIONAL Y LA VELOCIDAD RELATIVA. ESTA VISTA DEL HARDWARE ES LA TIPICA QUE DEBEN USAR LOS DISENADORES DE SISTEMAS OPERATIVOS ESPECIFICAMENTE EN EL MEDIO AMBIENTE DESARROLLADO EN EL CUAL LOS DISENOS DE HARDWARE PUEDEN SER MODIFICADOS.
EL APROVECHAMIENTO DEL HARDWARE EN LAS ULTIMAS 3 DECADAS HA SIDO MUCHO MAS DRAMATICO QUE EL APROVECHAMIENTO EN SOFTWARE.

ESTO ACTUALMENTE VIENE DE QUE LOS DISENADORES NO ESTAN DISPUESTOS A UTILIZAR UNA TECNOLOGIA PARTICULAR DE HARDWARE DADO QUE ELLOS ESPERAN A QUE UNA MEJOR TECNOLOGIA SEA DISPONIBLE TAN PRONTO COMO SEA POSIBLE. LOS DISENADORES DE SISTEMAS OPERATIVOS GENERALMENTE TIENEN POCAS OPCIONES EN ESTA CUESTION DADO QUE ELLOS DEBEN IMPLEMENTAR SISTEMAS OPERATIVOS CON TECNOLOGIAS EXISTENTES, ELLOS DEBEN DE TRATAR CON LA REALIDAD Y ECONOMIA DEL HARDWARE DE HOY.
ACTUALMENTE LOS ALMACENAMIENTOS ASOCIATIVOS Y CACHE SON MAS CAROS QUE EL ALMACENAMIENTO PRIMARIO DE ACCESO DIRECTO. ESTO NOS COMPROMETE A TENER UN MECANISMO DE MAPEO DE PAGINA.
UN ALMACENAMIENTO ASOCIATIVO CAPAZ DE SOSTENER SOLO UN PEQUENO PORCENTAJE DE MAPEO DE PAGINA COMPLETO ES USADO PARA UN PROCESO.
LA ENTRADA A LA PAGINA MANTENIDA EN ESTE MAPA CORRESPONDE SOLO A LAS PAGINAS MAS RECIENTEMENTE REFERENCIADAS. USANDO LA HEURISTICA DE COMO LA PAGINA SE REFERENCIA EN EL PASADO, ES COMO SERA REFERENCIADA EN EL FUTURO PROXIMO. LOS SISTEMAS DE HOY USAN UN MAPA DE PAGINA ASOCIATIVA PARCIAL QUE MEJORA LA EJECUCION DE UN 90% Y AUN MAS ALTO QUE LA POSIBLE EJECUCION CON UN MAPA DE PAGINA ASOCIATIVA COMPLETA.
LA TRANSICION DE DIRECCIONES DINAMICA PROCEDE COMO SIGUE. UN PROGRAMA EN EJECUCION REFERENCIA DIRECCIONES VIRTUALES v=(p,d) EL MECANISMO DE TRANSFORMACION DE DIRECCION PRIMERO TRATO DE ENCONTRAR LA PAGINA p EN EL MAPA PARCIAL DE PAGINA ASOCIATIVO. SI p ESTA EN EL MAPA ENTONCES EL MAPA ASOCIATIVO RETORNA A p', COMO EL NUMERO DE MARCO (FRAME) CORRESPONDIENTE A LA PAGINA VIRTUAL p Y SE HACE LA CONCATENACION DE p' CON EL DESPLAZAMIENTO d PARA FORMAR LA DIRECCION REAL r QUE CORRESPONDE A LA DIRECCION VIRTUAL v=(p,d). SI NO HAY UN PAR (MATCH) PARA LA PAGINA p EN EL MAPA DE LA PAGINA ASOCIATIVO UN MAPEO DIRECTO CONVENCIONAL ES USANDO LA DIRECCION b EN EL REGISTRO ORIGINAL DE LA TABLA DE PAGINA ES AGREGADO A p PARA LOCALIZAR UNA ENTRADA APROPIADA PARA LA PAGINA p, EN LA TABLA DE MAPEO DE PAGINA DE MAPEO DIRECTO EN ALMACENAMIENTO PRIMARIO.LA TABLA INDICA QUE p' ES LA PAGINA FRAME CORRESPONDIENTE A LA PAGINA VIRTUAL p,p' ES CONCATENADA CON EL DESPLAZAMIENTO d PARA FORMAR LA DIRECCION REAL r CORRESPONDIENTE A LA DIRECCION VIRTUAL v=(p,d). EL MAPEO PARCIAL DE PAGINA ASOCIATIVA NO NECESITA SER GRANDE PARA TENER UNA BUENA EJECUCION. DE HECHO, LOS SISTEMAS USAN TECNICAS QUE SOLO 8 O 16 REGISTROS DE ALMACENAMIENTO ASOCIATIVO FRECUENTEMENTE MEJORA EN UN 90% DE LA POSIBLE EJECUCION CON UN MAPEO ASOCIATIVO COMPLETO EN EL CUAL LOS SISTEMAS DE HOY TAL VEZ REQUIERAN 100 TIEMPOS CON MUCHAS ENTRADAS. ESTO OCURRE POR EL FENOMENO DE LOCALIDAD UNA PROPIEDAD DE UN PROCESO EN EJECUCION.
USAR UN MECANISMO DE MAPEO CON CONBINACION ASOCIATIVO Y DIRECTO ES UNA DESICION DE INGENIERIA BASADA EN LA ECONOMIA

DE LAS TECNOLOGIAS EXISTENTES DE HARDWARE. LAS TECNICAS DE HARDWARE CAMBIAN RAPIDAMENTE TAL QUE ES IMPORTANTE PARA LOS ESTUDIOS DE SISTEMAS OPERATIVOS CONOCER LAS TECNICAS DEL HARDWARE SURGIDOS.

7.9 COMPORTAMIENTO EN UN SISTEMA DE PAGINACION (SHARING)

EN SISTEMAS DE COMPUTADORA DE MULTIPROGRAMACION ESPECIALMENTE EN SISTEMAS DE TIEMPO COMPARTIDO ES COMUN PARA MUCHOS USARIOS EJECUTAR LOS MISMOS PROGRAMAS SI LA COPIA INDIVIDUAL DE ESOS PROGRAMAS FUERON DADOS A CADA USARIO, ENTONCES MUCHO DEL ALMACENAMIENTO PRIMARIO SERIA MAL UTILIZADO, LA SOLUCION OBVIA ES COMPARTIR ESAS PAGINAS.
EL COMPORTAMIENTO SE DEBE CONTROLAR CUIDADOSAMENTE PARA PREVENIR A UN PROCESO MODIFICAR DATOS QUE OTRO PROCESO ESTA LEYENDO. EN LA MAYOR PARTE DE LOS SISTEMAS DE HOY QUE IMPLEMENTAN LA COMPARTICION, (SHARING) LOS PROGRAMAS SON DIVIDIDOS DENTRO DE PROCEDIMIENTOS SEPARADOS Y AREAS DE DATOS. LOS PROCEDIMIENTOS NO MODIFICABLES SON LLAMADOS PROCEDIMIENTOS PUROS O REENTRANTES, LOS DATOS MODIFICABLES CLARAMENTE NO PODRAN SER COMPARTIBLES, LOS DATOS NO MODIFICABLES (TALES COMO INFORMACION TABULAR FIJA) PUEDEN SER COMPARTIDOS. TODOS ESTOS PUNTOS DE DISCUCION SON NECESITADOS PARA IDENTIFICAR CADA PAGINA COMO COMPARTIDA O NO COMPARTIDA, CADA PAGINA DE LOS PROCESOS HA SIDO CATEGORIZADA EN ESTA FORMA, ENTONCES EL COMPORTAMIENTO DE PAGINA PURA ES IMPLEMENTADO TENIENDO ENTRADA A LA TABLA DE MAPEO DE PAGINA DE DIFERENTES PROCESOS APUNTANDO A LA MISMA PAGINA FRAME, ENTONCES LA PAGINA FRAME ES COMPARTIDA ENTRE ESOS PROCESOS. EL COMPORTAMIENTO (SHARING) REDUCE LA CANTIDAD DE ALMACEANAMIENTO PRIMARIO NECESITADA POR UN GRUPO DE PROCESOS PARA SER EJECUTADA EFICIENTEMENTE Y PUEDE SER QUE UN SISTEMA DADO SOPORTE MAS USUARIOS. MAS ADELANTE LA EMISION DE COMO EL COMPORTAMIENTO AFECTA EL ALMACENAMIENTO VIRTUAL SERA CONSIDERADO.

7.10 SEGMENTACION

EN SECCIONES ANTERIORES DE ALMACENAMIENTO REAL NOSOTROS RECALCAMOS QUE EN MULTIPROGRAMACION EN LA VARIABLE DE PARTICION COLOCAMOS PROGRAMAS EN ALMACENAMIENTO QUE SON FRECUENTEMENTE EJECUTADOS SOBRE LAS BASES FIRST-FIT, BEST-FIT O WORDS-FIT. PERO NOSOTROS NOS LIMITAMOS A EJECUTAR PROGRAMAS EN BLOCKS DE LOCALIZACIONES CONTINUAS DE ALMACENAMIENTO REAL.
EN ESTE SISTEMA DE SEGMENTACION ESTA LIMITACION ES REMOVIDA Y UN PROGRAMA Y SUS DATOS SE LE PERMITE OCUPAR MUCHOS BLOCKS SEPARADOS CON ALMACENAMIENTO REAL. LOS BLOCKS DEL PROGRAMA

NO NECESITAN SER DEL MISMO TAMANO Y NO NECESITAN SER CONTINUOS. PERO LA SEPARACION DE LOS BLOCKS NECESITAN NO SER ADYACENTES A OTROS BLOCKS. ESTO INTRODUCE ALGUNOS PROBLEMAS INTERESANTES. POR EJEMPLO EL PROBLEMA DE PROTEJER A CADA USUARIO DE QUE SEA DESTRUIDO POR OTROS USUARIOS ESTO RESULTA SER MAS COMPLEJO. UN PAR DE REGISTROS FRONTERA NO NECESARIAMENTE GRANDE. SIMILARMENTE SE DIFICULTA EL LIMITE DE RANGO DE ACCESO DE ALGUNOS PROGRAMAS DADOS, UN ESQUEMA PARA LA IMPLEMENTACION DE PROTECCION DE ALMACENAMIENTO EN SISTEMAS DE SEGMENTACION ES EL USO DE LLAVES DE PROTECCION DE ALMACENAMIENTO. UNA DIRECCION VIRTUAL EN UN SISTEMA DE SEGMENTACION ES UN PAR ORDENADO v=(s,d) DONDE s ES IGUAL AL NUMERO DE SEGMENTO EN ALMACENAMIENTO VIRTUAL EN EL CUAL LOS ELEMENTOS REFERENCIADOS RECIDEN Y d ES EL DESPLAZAMIENTO CON SEGMENTO s, EL CUAL ES LA REFERENCIA QUE EL ELEMENTO ES LOCALIZADO. UN PROCESO SE PUEDE EJECUTAR SOLO SI ACTUALMENTE EL SEGMENTO ESTA EN ALMACENAMIENTO PRIMARIO. COMO UNIDADES COMPLETAS TODAS LAS LOCALIDADES ASIGNADAS A UN SEGMENTO SON COLOCADAS A LOCALIDADES CONTIGUAS EN ALMACENAMIENTO PRIMARIO. UN SEGMENTO ENTRANTE PUEDE SER COLOCADO EN ALGUN CONJUNTO DE LOCALIDADES CONTINUAS QUE ESTE DISPONIBLE EN ALMACENAMIENTO PRIMARIO Y QUE SEA LO SUFICIENTEMENTE LARGA PARA SOSTENER EL SEGMENTO. LAS ESTRATEGIAS PARA LA LOCALIZACION EN SEGMENTACION SON IDENTICAS A LAS QUE SE USAN EN VARIABLES DE PARTICION EN MULTIPROGRAMACION CON FIRST-FIT, BEST-FIT SIENDO LAS MAS COMUNES. LA TRANSICION DE DIRECCIONES DINAMICAS BAJO SEGMENTACION PROCEDE COMO SIGUE. UN PROCESO EN EJECUCION REFERENCIA UNA DIRECCION EN ALMACENAMIENTO VIRTUAL v=(s,d) EL MECANISMO DEL MAPEO DE SEGMENTO MIRA POR UN SEGMENTO s, EN LA TABLA DE MAPEO DE SEGMENTOS Y DETERMINA QUE EL SEGMENTO ESTA EN ALMACENAMIENTO REAL INICIANDO EN LA LOCALIDAD s'. LA DIRECCION DE ALMACENAMIENTO REAL CORRESPONDIENTE A LA DIRECCION DE ALMACENAMIENTO VIRTUAL v=(s,d) ES FORMADA POR LA CONCATENACION DE s' Y EL DESPLAZAMIENTO d. NOSOTROS EXAMINAMOS ESTE PROCESO MAS DETALLADAMENTE EN LA SIGUIENTES SECCIONES.

7.11 COMPORTAMIENTO (SHARING) EN UN SISTEMA DE SEGMENTACION

UNA DE LAS VENTAJAS DE SEGMENTACION SOBRE LA PAGINACION ES EL CONCEPTO LOGICO MAS BIEN QUE EL FISICO.
UN SEGMENTO QUE CORRESPONDE A UNA ESTRUCTURACION DE DATOS DINAMICOS PUEDE CRECER Y DISMINUIR EN TAMANO AL IGUAL QUE LOS DATOS. UN SEGMENTO QUE CORRESPONDE A UN CODIGO GENERADO POR UN COMPILADOR GRANDE SE NECESITA ESPERAR EL CODIGO.
LA SEGMENTACION SHARING ES SENCILLA CUANDO COMPARAMOS HACER EL SHARING EN SISTEMAS DE PAGINACION PURA. SI SE TIENE UN

ARREGLO EN SISTEMAS DE PAGINACION PURA DE 3.5 DE LONGITUD, EN ESE MOMENTO SE TIENE EN LA ENTRADA QUE INDICA QUE ES UN ARREGLO PERO SI SEPARAMOS LAS ENTRADAS PARA CADA PAGINA EN LA CUAL ESTA EL ARREGLO, DE TAL MODO DE QUE EL MANEJO DE LA PAGINA PARCIAL PUEDE SER DEFICIENTE.
LA SITUACION ES IGUAL O PEOR CON LA ESTRUCTURA DE DATOS DINAMICOS, YA QUE CUANDO ESTA CRECE EL TIEMPO DE EJECUCION SE HACE MAS LARGA.

¿ como se realiza LD

7.12 TRASFORMACION DE DIRECCIONAMIENTO SEGMENTADO POR EL METODO DIRECTO ?

EN LOS SISTEMAS DE PAGINACION SE TIENE QUE TENER ALGUNAS ESTRATEGIAS PARA LA IMPLEMENTACION DE LA TRASFORMACION DE DIRECCIONAMIENTO SEGMENTADO. ESTOS SISTEMAS DEBEN TENER YA SEA EL METODO DIRECTO, MODO ASOCIATIVO O LA CONBINACION DE ESTOS, ADEMAS DE TENER EL ALMACENAMIENTO CACHE DE TAL MANERA QUE ESPERA LA ENTRADA DEL SEGMENTO A LA TABLA DE MAP.
CONSIDERAREMOS EL CASO EN EL CUAL LA TRASFORMACION DIRECCIONADA PROCEDE NORMALMENTE, ALGUNOS PROBLEMAS Y COMO ATACARLOS. AL CORRER UN PROCESO HACIENDO REFERENCIA A DIRECCIONAMIENTO VIRTUAL EL NUMERO DE SEGMENTOS ES AGREGADA A LA BASE DIRECCIONADA Y LA TABLA MAP DEL SEGMENTO, ESTO ORIGINA QUE EL REGISTRO TENGA LA DIRECCION REAL ALMACENADA.
LA TABLA MAP DEL SEGMENTO CONTIENE LA DIRECCION EN ALMACENAMIENTO PRIMARIO EN DONDE COMIENZAN LOS SEGMENTOS.
SI UN SEGMENTO (s) ESTA OCURRIENDO EN ALMACENAMIENTO PRIMARIO ENTONCES LA DIRECCION DE ALMACENAMIENTO PRIMARIO ES (s'). SI NO ESTA EN ALMACENAMIENTO PRIMARIO ENTONCES (a) ES LA DIRECCION ALMACENADA DESDE EL CUAL EL SEGMENTO PUEDE SER LLAMADA ANTES DE SER EJECUTADA. TODAS LAS REFERENCIAS DEL SEGMENTO SON CHECADOS CONTINUAMENTE EN CUANTO A LA LONGITUD DEL SEGMENTO HASTA ESTAR SEGUROS DE QUE ESTA EN EL RANGO DE EL MISMO.

7.13 TRANSFORMACION DE DIRECCIONAMIENTO DINAMICO EN SISTENA DE PAGINACION/SEGMENTACION

AHORA CONSIDERAMOS LA TRASFORMACION DE DIRECCIONAMIENTO DINAMICO DE DIRECCIONAMIENTO VIRTUAL O REAL EN EL SISTEMA PAGINACION/SEGMENTACION USANDO COMBINACIONES DIRECTA/ ASOCIATIVA. PUEDE HABER PROCESOS DE TRASFORMACION QUE EN ALGUNA ETAPA ESPECIFICA EN QUE ESTA PUEDE FALLAR. UN MAPA DE SEGMENTO BUSCA EN UNA TABLA INDICADA AL SEGMENTO (s) QUE NO ESTA EN DIRECCIONAMIENTO PRIMARIO ESTO GENERA EL LLAMADO "MISSING SEGMENT FAULT" Y POR CONSECUENCIA CAUSA QUE EL S.O. A DE LOCALIZAR EL SEGMENTO EN EL ALMACENAMIENTO SECUNDARIO CREANDO DE ESTA MANERA UNA PAGINA PARA EL SEGMENTO Y LOCALIZA LA PAGINA APROPIADA EN EL ALMACENAMIENTO PRIMARIO,

¿ Diferencias entre direccionamiento segmentado y dinámico ?

AL HACER ESTO, POSIBLEMENTE REEMPLACE UNA PAGINA QUE YA EXISTE DE ESTE O ALGUNOS OTROS PROCESOS. SI EL SEGMENTO ESTA EN ALMACENAMIENTO PRIMARIO ENTONCES LA REFERENCIA DE LA PAGINA INDICARA QUE ESTA NO ESTE EN MEMORIA PRINCIPAL Y ESTO GENERA LO QUE SE CONOCE COMO " MISSING PAGE FAULT" CAUSANDO QUE EL S.O. CONTROLE NUEVAMENTE YA QUE LA LOCALIZA LA PAGINA EN ALMACENAMIENTO SECUNDARIO. EN UN DIRECCIONAMIENTO DE ALMACENAMIENTO VIRTUAL PARA PODER TERMINAR UN SEGMENTO SE GENERA EL LLAMADO "SEGMENT OVERFLOW FAULT" O YA SEA QUE LA PROTECCION DE BITS PUEDA SER INDICADA POR LA OPERACION A SER EJECUTADA Y QUE ESTA NO PUEDA ALMACENAR ENTONCES GENERA EL "SEGMENT PROTECTION FAULT".
EL ALMACENAMIENTO ASOCIATIVO TAMBIEN CONOCIDO COMO ALMACENAMIENTO CACHE RAPIDO ES CRITICO PARA EJECUTAR OPERACIONES EN LO REFERENTE A LOS MECANISMOS DE TRANSFORMACION DE DIRECCIONAMIENTO DIRECTO PARA EL COMPLETO MANTENIMIENTO DEL ALMACENAMIENTO PRIMARIO.
COMO REFERENCIA UN ITEM PUEDE VERSE ENVUELTO EN 3 CICLOS DE ALMACENAMIENTO Y LOS SISTEMAS COMPUTACIONALES DEBERIAN CORRER UNICAMETE 1/3 DE LA VELOCIDAD NORMAL, CON 2/3 ES EL TIEMPO GASTADO EN TRANSFORMACION DE DIRECCIONAMIENTO UNICAMENTE 8 O 16 REGISTROS ASOCIADOS, ALGUNOS SISTEMAS OPERATIVOS LLEGAN A TENER EFICIENCIA DEL 90% EN TIEMPO DE RESPUESTA Y MAS CUANDO EL PROCESAMIENTO ES RAPIDO.

7.14 COMPORTAMIENTO (SHARING) EN UN SISTEMA DE PAGINACION/SEGMENTACION

EN UNA PAGINACION/SEGMENTACION EL SEGMENTO SHARING ES IMPORTANTE YA QUE PERMITE TENER ENTRADAS EN UNA TABLA MAP POR DIFERENTES PUNTOS DEL PROCESO.
SIN EMBARGO EL SHARING PUEDE ESTAR EN SISTEMAS DE SEGMENTACION TENIENDO COMO CONSECUENCIA UN REQUERIMIENTO DE ADMINISTRACION CUIDADOSA DEL SISTEMA OPERATIVO.

ADMINISTRACION DE ALMACENAMIENTO VIRTUAL

8.1 INTRODUCCION

EN TEMAS ANTERIORES LAS DIFERENTES ORGANIZACIONES DE ALMACENAMIENTO VIRTUAL QUE HAN SIDO IMPLANTADAS FUERON DISCUTUDAS, ES DECIR: PAGINACION, SEGMENTACION, SEGMENTACION Y PAGINACION CONBINADAS. DISCUTIMOS EL HARDWARE Y EL MECANISMO DE SOFTWARE PARA IMPLEMENTAR ALMACENAMIENTO VIRTUAL, EN ESTE TEMA CONSIDERAMOS ESTRATEGIAS FARA MANEJAR SISTEMAS DE ALMACENAMIENTO VIRTUAL Y EXAMINAREMOS EL COMPORTAMIENTO DE SISTEMAS DE ALMACENAMIENTO VIRTUAL OPERANDO SOBRE ESAS ESTRATEGIAS.

8.2 ESTRATEGIAS DE TRABAJO DE ALMACENAMIENTO VIRTUAL

EN TEMAS ANTERIORES DISCUTIMOS ESTRATEGIAS DE ALMACENAMIENTO DE MANEJO FETCH,COLOCACION Y REEMPLAZO ESTAS SON RECONSIDERADAS AQUI EN EL CONTEXTO DE SISTEMAS DE ALMACENAMIENTO VIRTUAL.

- ESTRATEGIA FETCH: ESTAS CONCIERNEN COMO UNA PAGINA O UN SEGMENTO QUE PODRAN SER TRAIDOS DE ALMACENAMIENTO SECUNDARIO-PRIMARIO. LA DEMANDA DE LA ESTRATEGIA FETCH ESPERA POR UNA PAGINA O SEGMENTO A SER REFERENCIADA POR UN PROCESO CORRIENDO ANTES DE TRAER LA PAGINA O SEGMENTO A ALMACENAMIENTO PRIMARIO. ANTICIPADAMENTE EL ESQUEMA FETCH INTENTA DETERMINAR UN AVANCE CON PAGINAS O SEGMENTOS QUE SERAN REFERENCIADOS POR UN PROCESO. SI EL LIKELIHOOD DE REFERENCIA ES ALTO Y SI EL ESPACIO ESTA DISPONIBLE ENTONCES LA PAGINA O SEGMENTO PODRA SER TRAIDA DE ALMACENAMIENTO PRIMARIO ANTES DE QUE ESTA SEA EXPLICITAMENTE REFERENCIADA.
- ESTRATEGIA DE COLOCAION: CONCIERNE EN DONDE COLOCAR EN ALMACENAMIENTO PRIMARIO UNA PAGINA O SEGMENTO QUE VA A INGRESAR. EL SISTEMA PAGINA TRIVIALIZA LA DESICION DE COLOCACION DEBIDO A QUE UNA PAGINA QUE INGRESA FUEDE SER COLOCADA EN CUALQUIER MARCO DE PAGINA (FRAME PAGE) DISPONIBLE. SISTEMAS DE SEGMENTACION REQUIEREN ESTRATEGIAS DE COLOCACION SEMEJANTES A LAS DE SISTEMAS DE MULTIPROGRAMACION DE PARTICION VARIABLE.
- ESTRATEGIA DE REEMPLAZO: ESTO CONCIERNE CON DECIDIR CUALES PAGINAS O SEGMENTOS REMOVER PARA HACER LUGAR A UNA PAGINA O SEGMENTO QUE INGRESARA CUANDO EL ALMACENAMIENTO PRIMARIO YA ESTA COMPLETAMENTE ASIGNADO.

8.3 ESTRATEGIAS DE REEMPLAZO DE PAGINA

ESTO ES COMUNMENTE EN SISTEMAS CONPAGINADOS FARA TODAS LAS PAGINAS FRAME QUE ESTAN EN USO. EN ESTE CASO EL SISTEMA OPERATIVO MANEJARA LAS RUTINAS DE ALMACENAMIENTO QUE DEBERA

DECIDIR CUALES PAGINAS EN ALMACENAMIENTO PRIMARIO SERAN DESPLAZADAS PARA HACER LLEGAR UNA PAGINA QUE INGRESARA, CONSIDEREMOS CADA UNA DE LAS ESTRATEGIAS DE REEMPLAZO DE PAGINA:

- EL PRINCIPIO DE OPTIMALIDAD
- EL REEMPLAZO ALEATORIO DE PAGINA
- PRIMERA EN ENTRAR PRIMERA EN SALIR
- LA MENOS RECIENTEMENTE USADA
- LA MENOS FRECUENTEMENTE USADA
- NO USADA RECIENTEMENTE

8.4 EL PRINCIPIO DE OPTIMALIDAD

EL PRINCIPIO DE OPTIMALIDAD ES PARA OBTENER EJECUCION (PERFORMANCE) LA PAGINA REEMPLAZADA ES LA PRIMERA QUE NO SERA USADA OTRA VEZ EN UN TIEMPO NO MUY LEJANO DEL FUTURO. ES POSIBLE DEMOSTRAR LA OPTIMALIDAD DE ESA ESTRATEGIA PERO EL CURSO DE LA ESTRATEGIA NO ES REALIZABLE YA QUE NO PODEMOS PREDECIR EL FUTURO.
ASI PARA LOGRAR BUENA EJECUCION (PERFORMANCE) TRATAREMOS DE APROXIMAR EL PRINCIPIO DE OPTIMALIDAD POR EL USO DE TECNICA DE REEMPLAZO DE PAGINA QUE APROXIMADAMENTE REEMPLAZA LA PAGINA OPTIMA.

8.5 PAGINA ALEATORIAMENTE REEMPLAZADA

SI BUSCAMOS UN MENOR OVERHEAD EN LA ESTRATEGIA DE REEMPLAZO DE PAGINA QUE NO HAGA DISCRIMINACION CONTRA USUARIOS PARTICULARES, UNA SIMPLE TECNICA ES SELECCIONAR ALEATORIAMENTE LA PAGINA QUE SERA REEMPLAZADA, TODAS LAS PAGINAS EN ALMACENAMIENTO PRINCIPAL ASI TENDRAN UN LIKELIHOOD IGUAL DE SER SELECCIONADAS PARA REEMPLAZARLAS ESTA ESTRATEGIA PODRA SELECCIONAR CUALQUIER PAGINA PARA REEMPLAZARLA INCLUYENDO LA SIGUIENTE PAGINA HA SER REFERENCIADA (LA CUAL ES DE CURSO LA PEOR PAGINA PARA REEMPLAZAR) DEBIDIO A QUE ES APROXIMADAMENTE ACERTAR O NO ACERTAR, ESTE ESQUEMA ES RARAMENTE USADO.

8.6 PAGINA REMPLAZADA PRIMERA-ENTRAR-PRIMERA-SALIR (FIFO)

EN LA PAGINA REEMPLAZDA FIFO MARCAREMOS EL TIEMPO DE CADA PAGINA SEGUN ESTE ENTRANDO A ALMACENAMIENTO PRIMARIO. CUANDO UNA PAGINA NECESITE SER REEMPLAZADA SELECCIONAREMOS LA PRIMERA QUE TUVO UN MAYOR TIEMPO EN ALMACENAMIENTO. LA ATRACCION INTUITIVA DE ESTA ESTRATEGIA PARECE RAZONABLE, ES DECIR QUE ESA PAGINA TENIA ESA OPORTUNIDAD Y ESTA VEZ ES DADA A OTRA PAGINA.
DESAFORTUNADAMENTE, FIFO ES PROBABLEMENTE TRABAJOSO REEMPLAZAR PAGINAS USADAS PORQUE LA RAZON DE QUE UNA PAGINA

ESTUVO EN ALMACENAMIENTO PRIMARIO POR UN LARGO TIEMPO PODRIA SER BIEN QUE ESTABA EN CONSTANTE USO. POR EJEMPLO EN SISTEMAS LARGOS DE TIEMPO COMPARTIDO (TIMESHARING) ESTO ES COMUN PARA MUCHOS USUARIOS COMPARTIR UNA COPIA DE UN EDITOR DE TEXTOS SEGUN TIPO Y PROGRAMAS CORRECTOS. REEMPLAZO DE PAGINAS FIFO EN TAL SISTEMA PODRIA ESCOGER UN USO PESADO DEL EDITOR PARA REEMPLAZAR PAGINAS, ESTO SERIA CIERTAMENTE LA ELECCION. ESTA PAGINA SERIA RECLAMADA A ALMACENAMIENTO PRINCIPAL CASI INMEDIATAMENTE.

8.7 ANOMALIA FIFO

PARECERIA RAZONABLE QUE MAS MARCOS DE PAGINAS SE LE ASIGNARAN A UN PROCESO EL EXPERIMENTA FEWER PAGINA FAULTS. BELADY,NELSON Y SHARDLER DESCUBRIERON QUE DENTRO DE REEMPLAZO DE PAGINA FIFO, CIERTOS PATRONES DE PAGINAS REFERENCIADAS ACTUALMENTE CAUSAN MAS PAGINAS FAULTS CUANDO EL NUMERO DE PAGINAS FRAME ASIGNADAS A UN PROCESO ESTA EN AUMENTO, ESTE ES EL FENOMENO LLAMADO ANOMALIA FIFO.
LA ANOMALIA FIFO ESTA CONSIDERADA COMO UNA CURIOSIDAD MAS QUE UN IMPORTANTE RESULTADO. A CASO ESTA ES SIGNIFICANCIA REAL PARA EL ESTUDIANTE DE S.O., ESTO SIRVE COMO UNA PRECAUCION QUE LOS S.O. SON ENTIDADES COMFLEJAS QUE ALGUNAS VECES RETAN A LA INTUICION.

8.8 REEMPLAZO DE PAGINA MENOR-RECIENTEMENTE-USADA (LRU)

ESTA ESTRATEGIA SELECCIONA LA PAGINA QUE NO FUE USADA POR UN LARGO TIEMPO PARA REEMPLAZARLA. AQUI CONTAMOS CON LA HEURISTICA EN LA QUE EL PASADO RECIENTE ES BUEN INDICADOR DEL FUTURO CERCANO. LRU REQUIERE QUE CADA PAGINA ESTE EN SU TIEMPO ESTAMPADO (TIME STAMPED) CADA VEZ QUE ES REFERENCIADA ESTA REQUIERE SUSTANCIAL OVERHEAD Y ASI LA ESTRATEGIA LRU SI BIEN ATRAE NI ES A MENUDO IMPLEMENTADA EN SISTEMAS COMUNES, EN LUGAR DE ESO ESTRATEGIAS CON MENOR OVERHEAD QUE SE APROXIMAN A LRU SON USADAS. DEBEMOS SER CUIDADOSOS CUANDO APLIQUEMOS RAZONAMIENTO HEURISTICO EN S.O. POR EJEMPLO EN LA ESTRATEGIA LRU, LA PAGINA RECIENTEMENTE MENOS USADA PUDO DE HECHO SER LA SIGUIENTE PARA SER USADA COMO UN PROGRAMA CICLADO ESTA ES UNA MANERA EN TURNO A UN LARGO CICLO ENVOLVIENDO VARIAS PAGINAS. SEGUN EL REEMPLAZO DE PAGINA LRU NOSOTROS MISMOS ENCONTRAREMOS QUE PAGINADO ES ADECUADO RESPALDAR DENTRO DE ALMACENAMIENTO PRINCIPAL CASI INMEDIATAMENTE.

8.9 REEMPLAZO DE PAGINA MENOR-FRECUENTEMENTE-USADA (LFU)

UNA APROXIMACION A LRU ES LA ESTRATEGIA MENOR-FRECUENTEMENTE-USADA (LFU). AQUI ESTAMOS INTERESADOS EN

COMO FUE INTENSAMENTE EL USO DE CADA PAGINA. LA PAGINA A REEMPLAZAR ES AQUELLA PAGINA QUE ES MENOR FRECUENTEMENTE USADA O MENOR INTENSAMENTE REFERENCIADAS. AQUI NUEVAMENTE LA ATRACCION INTUITIVA ES REAL. SIN EMBARGO, UNA PAGINA MALA PUDO DEL TODO FACILMENTE SER SELECCIONADA PARA REEMPLAZARLA, POR EJEMPLO LA PAGINA MENOR FRECUENTEMENTE USADA PUDO SER LA PAGINA TRAIDA DENTRO DE ALMACENAMIENTO PRINCIPAL MAS RECIENTEMENTE. LA PAGINA FUE USADA UNA VEZ MIENTRAS QUE TODAS LAS OTRAS PAGINAS EN ALMACENAMIENTO PRIMARIO PODRIAN SER USADAS MAS DE UNA VEZ. NUNCA EL MECANISMO DE REEMPLAZO DE PAGINA REEMPLAZARA ESTA PAGINA CUANDO EN RELACION ESTA SEA ALTAMENTE PROBABLE SER USADA INMEDIATAMENTE.
CIERTAMENTE PARECE COMO PRECAUCION DE QUE CADA ESQUEMA DE REEMPLAZO DE PAGINA CORRE ALGUN PELIGRO DE TOMAR MALAS DESICIONES. ESTO ES VERDAD CIERTAMENTE DEBIDO A QUE NO PODEMOS PRECISAMENTE PREDECIR EL FUTURO. UNA ESTRATEGIA DE REEMPLAZO DE PAGINA QUE HACE MAS DESICIONES RAZONABLES DE TIEMPO TIENE MENOR OVERHEAD DESEADO.

8.10 REEMPLAZO DE PAGINA NO-RECIENTEMENTE-USADA (NUR)

UN POPULAR ESQUEMA PARA APROXIMAR LRU CON PEQUENO OVERHEAD ES NO-RECIENTEMENTE-USADA (NUR). PAGINAS NO USADAS RECIENTEMENTE NO TIENEN PROBABILIDAD DE SER USADAS EN UN FUTURO PROXIMO Y DEBEN SER REEMPLAZADAS CON INGRESO DE PAGINAS. DEBIDO A ESTO ES DESEABLE REEMPLAZAR UNA PAGINA QUE NO FUE CAMBIADA MIENTRAS ESTUVO EN ALMACENAMIENTO PRIMARIO, LA ESTRATEGIA NUR IMPLEMENTADA CON LA ADICION DE 2 BITS DE HARDWARE POR PAGINA ESTOS SON:

BIT REFERENCIADO = 0 SI LA PAGINA NO FUE REFERENCIADA
= 1 SI LA PAGINA FUE REFERENCIADA

BIT MODIFICADO = 0 SI LA PAGINA NO FUE MODIFICADA
= 1 SI LA PAGINA FUE MODIFICADA

EL BIT MODIFICADO ES MUCHAS VECES LLAMADO BIT DIRTY.

LA ESTRATEGIA NUR TRABAJA COMO SIGUE, INICIALMENTE LOS BITS REFERENCIADOS DE TODAS LAS PAGINAS ESTAN PUESTOS EN 0. CUANDO OCURRA UNA REFERENCIA A UNA PAGINA EN PARTICULAR EL BIT REFERENCIADO DE ESA PAGINA ES PUESTO EN 1. INICIALMENMTE. LOS BITS MODIFICADOS EN TODAS LAS PAGINAS SERAN 0. SIN EMBARGO UNA PAGINA ES MODIFICADA. ESTE BIT MODIFICADO ES PUESTO EN 1. CUANDO UNA PAGINA ES REEMPLAZADA PRIMERO INTENTAREMOS ENCONTRAR UNA PAGINA LA CUAL FUE REFERENCIADA (DEBIDO A QUE ESTAMOS APROXIMANDO A LRU). DE OTRO MODO NOTEMOS QUE SELECCIONAMOS PERO REEMPLAZAMOS LA PAGINA REFERENCIADA, SI UNA PAGINA NO FUE REFERENCIADA CHECAMOS SI ESTA PAGINA FUE O NO MODIFICADA, REEMPLAZAMOS

ESTA CON LA BASE DE QUE HAY MENOR OVERHEAD INVOLUCRADO QUE EN REEMPLAZAR LA PAGINA MODIFICADA QUE DEBIO SER ESCRITA POSTERIORMENTE EN ALMACENAMIENTO SECUNDARIO. DE OTRO MODO DEBEMOS REEMPLAZAR UNA PAGINA MODIFICADA. EL ALMACENAMIENTO PRINCIPAL SERA ACTIVAMENTE REFERENCIADO EN SISTEMAS DE MULTIPLES USUARIOS Y MAS PRONTO O MAS TARDE LOS BITS REFERENCIADOS SERAN ENCENDIDOS, ENTONCES PERDEREMOS NUESTRA HABILIDAD PARA DISTINGUIR LA PAGINA MAS DESEABLE PARA REEMPLAZAR UNA TECNICA QUE FUE AMPLIAMENTE IMPLANTADA PARA EVITAR ESTE PROBLEMA, ES PERIODICAMENTE PONER TODOS LOS BITS REFERENCIADOS EN O PARA CONSEGUIR UN NUEVO INICIO Y ENTONCES ASIGNAMOS A LOS BITS REFERENCIADOS SER PUESTOS A 1. OTRA VEZ REFERENCIANDO BAJO MODELOS NORMALES ESTO HACE IGUALAR LAS PAGINAS ACTIVAS VULNERABLES PARA REEMPLAZO, PERO SOLAMENTE POR UN BREVE MOMENTO DESPUES LOS BITS SON REPUESTOS. PAGINAS ACTIVAS TENDRAN SUS BITS REFERENCIADOS PUESTOS A 1 OTRA VEZ INMEDIATAMENTE. EL NUR COMO SE DESCRIBIO ANTERIORMENTE RESULTA DENTRO DE LA EXISTENCIA DE 4 GRUPOS DE PAGINAS:

- GRUPO 1 NO MODIFICADAS NO REFERENCIADAS
- GRUPO 2 NO MODIFICADAS REFERENCIADAS
- GRUPO 3 REFERENCIADAS NO MODIFICADAS
- GRUPO 4 REFERENCIADAS MODIFICADAS

LAS PAGINAS NUMERADAS EN LOS GRUPOS MAS BAJOS SERIAN REEMPLZADAS PRIMERO Y AQUELLAS PAGINAS NUMERADAS EN LOS GRUPOS MAS ALTOS SERIAN REEMPLAZADAS DESPUES. NOTE QUE EL GRUPO 2 PARECE DESCRIBIR UNA SITUACION NO REALISTA ES DECIR PAGINAS QUE FUERON MODIFICADAS PERO NO REFERENCIADAS. ACTUALMENTE ESTA ES UNA SIMPLE CONSECUENCIA DE LA PERIODICA REPOSICION DE LOS BITS REFERENCIADOS (PERO NO DE LOS BITS MODIFICADOS) Y ES PERFECTAMENTE RAZONABLE.

8.11 LOCALIDAD

EL CENTRO PARA LAS MAYORIAS DE LAS ESTRATEGIAS DE MANEJO ES EL CONCEPTO DE LOCALIDAD, ESTE PROCESO TIENDE A REFERIRSE AL ALMACENAMIENTO NO UNIFORME, EN PATRONES ALTAMENTE LOCALIZADOS. LA LOCALIDAD SE MANIFIESTA TANTO EN EL TIEMPO COMO EN EL ESPACIO. LA LOCALIDAD TEMPORAL ES LA QUE SE REFIERE AL TIEMPO (OVERTIME) POR EJEMPLO, SI EL CLIMA ES CALIDO A LAS 3 PM ENTONCES ES UNA BUENA OPORTUNIDAD (AUNQUE NO GARANTIZADO) QUE EL TIEMPO ESTUVO SOLEADO A LAS 2.5 PM Y PODRA SEGUIR IGUAL A LAS 3.5 PM. LA LOCALIDAD ESPACIAL SIGNIFICA QUE LOS PUNTOS MAS CERCANOS TIENDEN A SER SIMILARES. DE NUEVO CONSIDERANDO EL TIEMPO, SI ESTA SOLEADO EN UNA CIUDAD ENTONCES ES PROBABLE (PERO NO GARANTIZABLE) QUE EN OTRAS CIUDADES CERCANAS ESTARA IGUAL. LA LOCALIDAD TAMBIEN SE OBSERVA EN AMBIENTES DE SISTEMAS DE OPERACION, PARTICULARMENTE EN EL AREA DEL MANEJO DE ALMACENAMIENTO. ESTO ES UNA PROPIEDAD EMPIRICA (SE OBSERVO) EN LUGAR DE UNA

TEORICA. ESTO NUNCA SE GARANTIZA PERO CON FRECUENCIA ES MUY PROBABLE. POR EJEMPLO EN SISTEMAS DE PAGINACION, OBSERvAMOS QUE LOS PROCESOS TIENDEN A FAVORECER CIERTOS GRUPOS DE LAS PAGINAS Y ESTAS CON FRECUENCIA TIENDEN A SER ADYACENTES EN UN ESPACIO DE DIRECCION VIRTUAL EN UN PROCESO. ESTO NO SIGNIFICA QUE UN PROCESO NO HARA REFERENCIA A LA NUEVA PAGINA SI ESTE FUESE EL CASO ENTONCES LOS PROCESOS NO PODRIAN EMPEZAR A CORRERSE EN PRIMER LUGAR. ESTO SIGNIFICA QUE UN PROCESO TERMINARA A CONCENTRAR SUS REFERENCIAS EN UN INTERVALO DE TIEMPO A UN SUBCONJUNTO PARTICULAR DE SUS PAGINAS. DE HECHO, LA LOCALIDAD ES COMPLETAMENTE RAZONABLE EN UN SISTEMA DE COMPUTADORAS CUANDO SE CONSIDERA LA MANERA EN QUE SE ESCRIBEN LOS PROGRAMAS Y COMO SE ORGANIZAN LOS DATOS. EN RESUMEN:

1.- LA LOCALIDAD TEMPORAL SIGNIFICA QUE LAS LOCALIZACIONES DE ALMACENAMIENTO REFERIDAS ES MUY PROBABLE QUE SE VUELVAN A UTILIZAR EN UNA FUTURO CERCANO. PARA APOYAR ESTA OBSERVACION TENEMOS:

a) ASEGURANDO
b) SUBRUTINAS
c) PILAS
d) VARIABLES USADAS PARA CONTAR Y TOTALIZAR

2.- LOCALIDAD ESPACIAL SIGNIFICA QUE LAS REFERENCIAS DE ALMACENAMIENTO TIENDEN A SER AGRUPADAS DE TAL FORMA QUE UNA VEZ QUE SE HACE REFERENCIA A UNA LOCALIZACION ES MUY PROBABLE QUE LAS POSICIONES CERCANAS PODRAN SER REFERIDAS. PARA APOYAR ESTA OBSERVACION TENEMOS:

a) ORGANIZACION DE TRANSVERSALES
b) EJECUCION DEL CODIGO SECUENCIAL
c) LA TENDENCIA DE LOS PROGRAMADORES DE COLOCAR LAS DEFINICIONES DE LAS VARIABLES RELACIONADAS CERCA UNA DE OTRA.

TAL VEZ LA CONSECUENCIA MAS SIGNIFICATIVA DE LA LOCALIDAD DE REFERENCIA DE ALMACENAMIENTO ES QUE UN PROGRAMA PUEDE CORRERSE EFICIENTEMENTE A MEDIDA QUE EL SUBCONJUNTO DE PAGINAS ELEGIDO ESTE EN EL ALMACENAMIENTO PRIMARIO.

8.12 CONJUNTOS DE TRABAJOS

DENNING DESARROLLO UN ENFOQUE DEL PROGRAMA DE PAGINACION LLAMADO LA TEORIA DEL CONJUNTO DE TRABAJO DE LA CONDUCTA DEL PROGRAMA. INFORMALMENTE UN CONJUNTO DE TRABAJO ES UNA COLECCION DE PAGINAS A LAS QUE EL PROCESO HACE REFERENCIAS INFORMATIVA. DENNING SOSTUVO QUE PARA QUE UN PROGRAMA CORRA EFICIENTEMENTE EL CONJUNTO DE TRABAJO DE LAS PAGINAS DEBEN MANTENERSE EN EL ALMACENAMIENTO PRIMARIO DE OTRA MANERA. LA ACTIVIDAD DE PAGINACION EXCESIVA LLAMADA BASURA (MALA CALIDAD) PODRIA PRESENTARSE A MEDIDA QUE EL PROGRAMA SOLICITA EN FORMA REPETITIVA PAGINAS PARA EL

ALMACENAMIENTO SECUNDARIO.

UNA POLITICA DE MANEJO DE ALMACENAMIENTO DE CONJUNTO DE TRABAJO BUSCA MANTENER ESTOS CONJUNTOS DE LOS PROGRAMAS ACTIVOS EN EL ALMACENAMIENTO PRIMARIO. LA DESICION PARA ANADIR UN NUEVO PROCESO AL CONJUNTO ACTIVO DE PROCESOS (ES DECIR, INCREMENTAR EL NIVEL DE MULTIPROGRAMACION) ESTA BASADO EN DISPONIBILIDAD DEL ESPACIO SUFICIENTE DEL ALMACENAMIENTO PRIMARIO PARA ACOMODAR EL CONJUNTO DE TRABAJO DE LAS PAGINAS DEL PROCESO NUEVO. ESTA DESICION ESPECIALMENTE EN EL CASO DE PROCESOS INICIADOS RECIENTEMENTE SE HACE CON FRECUENCIA CON EL USO DE HEURISTICOS, YA QUE ES IMPOSIBLE PARA EL SISTEMA CONOCER CON ANTERIORIDAD QUE TAN GRANDE SERA UN CONJUNTO DE TRABAJO DE UN PROCESO DADO.

EL CONJUNTO DE TRABAJO DE LAS PAGINAS DE UN PROCESO W(t,w) EN EL TIEMPO t, ES EL CONJUNTO DE PAGINAS REFERIDAS POR EL PROCESO DURANTE EL INTERVALO DE TIEMPO DEL PROCESO t-w A t. EL TIEMPO DE PROCESO ES EL TIEMPO DURANTE EL CUAL UN PROCESO TIENE AL CPU. LA VARIABLE W ES LLAMADA "WORKING SET WINDOW SIZE" Y LA DETERMINACION DE QUE TAN GRANDE DEBE SER W ES CRITICA A LA OPERACION EFECTIVA DE UNA ESTRATEGIA DE MANEJO DE ALMACENAMIENTO DE TRABAJO. CUANDO SE INCREMENTA LA MEDIDA DEL CONJUNTO DE TRABAJO AL TIEMPO QUE W SE INCREMENTA, ESTA ES UNA CONSECUENCIA DE LA DEFINICION MATEMATICA DEL CONJUNTO DE TRABAJO Y NO ES UNA INDICACION DEL TAMANO DEL CONJUNTO DE TRABAJO OBSERVABLE EMPIRICAMENTE. EL CONJUNTO DE TRABAJO REAL DE UN PROCESO ES EL GRUPO DE PAGINAS QUE DEBEN ESTAR EN EL ALMACENAMIENTO PRIMARIO PARA QUE UN PROCESO EJECUTE EFICIENTEMENTE.

LOS CONJUNTOS DE TRABAJO CAMBIAN A MEDIDA QUE SE LLEVA A CABO EL PROCESO. ALGUNAS VECES LOS CAMBIOS DRAMATICOS SE PRESENTAN CUANDO EL PROCESO ENTRA EN UNA FASE DE EJECUCION QUE REQUIERE UN CONJUNTO DE TRABAJO COMPLETAMENTE DIFERENTE. ASI CUALQUIER ASEVERACION ACERCA DEL TRABAJO Y CONTENIDO DEL CONJUNTO DE TRABAJO INICIAL DEL PROCESO, NO SE APLICA NECESARIAMENTE A LOS CONJUNTOS DE TRABAJOS SUBSECUENTES QUE EL PROCESO ACUMULARA. ESTO COMPLICA EL MANEJO DEL ALMACENAMIENTO PRECISO, BAJO UNA ESTRATEGIA DE TRABAJO. UNA VEZ QUE EL PROCESO REQUIERE PAGINAS EN SU CONJUNTO DE TRABAJO (UNA PAGINA A LA VEZ, EL PROCESO GRADUALMENTE RECIBE SUFICIENTE ALMACENAMIENTO PARA SOSTENER SU CONJUNTO DE TRABAJO). AL LLEGAR A ESTE PUNTO, EL USO DEL ALMACENAMIENTO ESTABILIZA A MEDIDA QUE HACE REFERENCIA ACTIVA A LAS PAGINAS DE SU PRIMER CONJUNTO DE TRABAJO. EVENTUALMENTE EL PROCESO HARA UNA TRANSICION AL SIGUIENTE CONJUNTO DE TRABAJO. EL SISTEMA NO TIENE FORMA DE SABER SI EL PROCESO EXPANDE EL CONJUNTO DE TRABAJO O SOLO SE CAMBIA. UNA VEZ QUE EL PROCESO SE ESTABILIZA EN EL SIGUIENTE CONJUNTO DE TRABAJO, EL SISTEMA CONSIDERA MENOS REFERENCIAS DE LA PAGINA EN LA VENTANA Y REDUCE LA ASIGNACION DEL ALMACENAMIENTO PRIMARIO

DEL PROCESO AL NUMERO DE PAGINAS, AL SEGUNDO CONJUNTO DE TRABAJO. CADA VEZ QUE SE PRESENTA UNA TRANSICION ENTRE LOS CONJUNTOS DE TRABAJOS UNA LINEA CURVA QUE SUBE Y BAJA MUESTRA COMO EL SISTEMA SE ADAPTA AL LA TRANSICION.

8.13 PAGINACION DE DEMANDA

ES EL CONOCIMIENTO CONVENCIONAL QUE LAS PAGINAS DEL PROCESO DEBERIAN DE CARGAR SOBRE LA DEMANDA. NINGUNA PAGINA DEBERA TRAERSE DE UN ALMACENAMIENTO SECUNDARIO A UN PRIMARIO HASTA QUE SU EXPLICIDAD SEA REFERIDO POR UN PROCESO CORRIDO. EXISTEN VARIAS RAZONES PARA LA APELACION DE ESTA ESTRATEGIA. LOS RESULTADOS COMPUTABILIZADOS ESPECIFICAMENTE EL PROBLEMA CLAUDICADO NOS DICE QUE LA RUTA DE EJECUCION QUE UN PROGRAMA TOMARA PUEDE NO SER PREDECIDO CON PRESICION, POR LO TANTO CUALQUIER INTENTO PARA RECARGAR PAGINAS EN ANTICIPACION A SU USO, PUEDE RESULTAR SER CARGADO EN LAS PAGINAS EQUIVOCADAS. LA PAGINACION DE DEMANDA GARANTIZA QUE LAS UNICAS PAGINAS TRAIDAS AL ALMACENAMIENTO PRINCIPAL SON AQUELLAS QUE EN REALIDAD NECESITAN SER PROCESADAS. LA LINEA MAS ALTA INVOLUCRADA EN DECIDIR CUALES PAGINAS VAN AL ALMACENAMIENTO PRINCIPAL ES MINIMO. LAS ESTRATEGIAS QUE ANTICIPAN PARA TOMAR PAGINAS PUEDE REQUERIR SOBRE TIEMPO DE EJECUCION SUBSTANCIAL SOBRE LA LINEA QUE TANTAS PAGINAS ESTAN YA EN MEMORIA PRINCIPAL, ESTE ESPERA VOLVERSE CADA VEZ MAS COSTOSO COMO MAYOR CANTIDAD DE ALMACENAMIENTO ES OCUPADA POR PROCESOS EN ESPERA. REDUCIENDO EL TIEMPO DE ESPACIO (SPACE TIME PRODUCT) DE LAS PAGINAS DE UN PROCESO EN ESPERA ES UN OBJETIVO IMPORTANTE DE ESTRATEGIAS DE ADMINISTRACION DE ALMACENAMIENTO.

8.14 ANTICIPANDO PAGINAS (ANTICIPATORY PAGING)

UN TEMA CENTRAL EN LA ADMINISTRACION DE RECURSOS ES EL VALOR RELATIVO DE LA INFLUENCIA DE UN RECURSO JUSTO CON LA INTENSIDAD DEL RECURSO A SER MANEJADO. COSTOS EN HARDWARE VAN DECRECIENDO DRAMATICAMENTE. EL VALOR RELATIVO DE TIEMPO DE MAQUINA CON EL TIEMPO DE PERSONAS SE REDUJO CONSIDERABLEMENTE. AHORA DISENOS DE S.O. SON RELACIONADOS CON METODOS DE REDUCCION DE LA CANTIDAD DE TIEMPO DE PERSONA QUE DEBE ESPERAR RESULTADOS DE UN COMPUTADOR. ANTICIPATORY PAGING ES UNA TECNICA QUE TIENE PROMESA.

EN ANTICIPATORY PAGING, EL S.O. INTENTA PREDECIR LAS PAGINAS QUE UN PROCESO VA A NECESITAR Y ENTONCES PRECARGA ESAS PAGINAS CUANDO HAY ESPACIO DISPONIBLE. SI LA DESICION ES CORRECTA, EL TIEMPO TOTAL DE CORRIDA DEL PROCESO PUEDE SER REDUCIDO CONSIDERABLEMENTE. MIENTRAS EL PROCESO CORRE CON ESAS PAGINAS ACTUALES, EL SISTEMA CARGA NUEVAS PAGINAS QUE ESTARAN DISPONIBLES CUANDO LAS SOLICITE EL PROCESO.

ANTICIPATORY PAGING OFRECE LAS SIGUIENTES VENTAJAS:

- SI LA DESICION CORRECTA PUEDE SER TOMADA EN LA GRAN MAYORIA DE LOS CASOS, EL TIEMPO DE CORRIDA DE UN PROCESO DEBE SER REDUCIDO CONSIDERABLEMENTE. POR LO TANTO, ESTE ES DIFICIL MERECEDOR DE DESARROLLO DE MECANISMOS DE ANTICIPATORY PAGING AUN CUANDO NO PUEDEN SER 100% EXACTOS.
- DESICIONES EXACTAS PUEDEN SER HECHAS EN MUCHOS CASOS. SI ESTA DESICION HECHA PUEDE SER IMPLEMENTADA CON BAJO OVERHEAD, ENTONCES LA EJECUCION DE UN PROCESO DADO PUEDE SER CONSIDERABLEMENTE RAPIDO CONTRARIAMENTE FUERA DE OTROS CONMOVEDORES PROCESOS ACTIVOS.
- CON UN APROPIADO HARDWARE MAS ECONOMICO LAS CONSECUENCIAS DE UNA MALA DESICION SON MENOS SERIAS. PODEMOS COMPRAR MAYOR ALMACENAMIENTO EXTRA PARA SOPORTAR LA ACUMULACION DE EL EXCESO DE PAGINAS Y UN MECANISMO PREVISOR LAS LLEVARIA AL ALMACENAMIENTO PRIMARIO.

8.15 LIBERACION DE PAGINA

TRABAJANDO BAJO EL CONJUNTO DE MANEJO DE ALMACENAMIENTO LOS PROGRAMADORES NOS DICEN QUE PAGINA QUIEREN USAR POR REFERENCIA EXPLICITA. PROGRAMAS TAN LARGOS NECESITAN ESPECIFICAR SUS PAGINAS PARA PASAR AQUELLAS PAGINAS DESDE SU CONJUNTO DE TRABAJO. USUALMENTE HAY UN PERIODO DE DESICION DE TIEMPO EN EL CUAL LAS PAGINAS NO LARGAS NECESITAN PERMANECER EN ALMACENAMIENTO PRINCIPAL.

CUANDO QUEDA CLARO QUE UNA PAGINA NO LARGA SE NECESITARA, UN USUARIO PODRIA RESULTAR UNA LIBERACION DE PAGINA VOLUNTARIA PARA LIBERAR EL MARCO DE LA PAGINA. ESTO ELIMINARIA EL RETRASO DE PERIODOS CAUSADO DEJANDO EL PROCESO GRADUALMENTE PASAR LA PAGINA DESDE SU CONJUNTO DE TRABAJO.

LIBERACION DE PAGINA VOLUNTARIA ELIMINARIA DESGASTE Y RAPIDEZ DE EJECUCION DEL PROGRAMA. LA MAYOR PARTE DE LOS USUARIOS DE SISTEMAS COMPUTACIONALES ACTUALES, NO ESTAN ENTERADOS DE LO QUE ES UNA PAGINA, ESOS USUARIOS NO SABEN HACER DESICIONES DE ETIQUETAS DEL SISTEMA (SISTEM-LEVEL).

LA INCORPORACION DE COMANDOS DE LIBERACION DE PAGINA DENTRO DE LOS PRGRAMAS DE USUARIO AUMENTA CONSIDERABLEMENTE EL LENTO DESARROLLO DE APLICACIONES. LA ESPERANZA REAL EN ESTA AREA PARA COMPILADORES Y S.O. ES DETECTAR AUTOMATICAMENTE SITUACIONES DE LIBERACION DE PAGINA Y MUCHO MAS POSIBLES ESTRATEGIAS SOBRE CONJUNTOS DE TRABAJO.

8.16 TAMANO DE PAGINA

EN SISTEMAS DE PAGINACION, ALMACENAMIENTO REAL ES NORMALMENTE DIVIDIDO DENTRO DE MARCOS DE PAGINA DE TAMANO ARREGLADO. QUE TAN LARGO DEBE SER ESE MARCO DE PAGINA ? QUE TAN LARGA ES UNA PAGINA ? TODOS LAS PAGINAS DE UN SISTEMA

SON DEL MISMO TAMANO O VARIAN LOS TAMANOS DE PAGINAS USADOS? SI DIVERSOS TAMANOS SON USADOS EL TAMANO DE LAS PAGINAS GRANDES ES MULTIPLO DEL TAMANO DE LAS PEQUENAS ? ESAS PREGUNTAS REQUIEREN UN JUICIO Y UN CUIDADOSO ENTENDIMIENTO DEL HARDWARE, SOFTWARE Y UN PROMEDIO DE APLICACIONES PARA UN SISTEMA EN PRTICULAR. NO HAY RESPUESTAS UNIVERSALES, NO HAY URGENTE NECESIDAD PARA TODOS LOS SISTEMAS COMPUTACIONALES PARA TENER EL MISMO TAMANO DE PAGINA SIMPLE.

CONSIDERACIONES QUE DETERMINAN SI UNA PAGINA SERA LARGA O NO:

- EL CORTO TAMANO DE LA PAGINA, LAS PAGINAS Y LOS MARCOS DE PAGINAS QUE HAY Y EL LARGO NECESITAN ESTAR EN LA TABLA DE PAGINAS. SISTEMAS EN LOS CUALES LAS TABLAS DE PAGINAS OCUPAN ALMACENAMIENTO PRIMARIO SON PUNTOS PARA LA NECESIDAD DE PAGINAS LARGAS. EL DERROCHE DE ALMACENAMIENTO DEBIDO A LAS TABLAS EXCESIVAMENTE LARGAS ES LLAMADO FRAGMENTACION DE TABLAS. NOTAMOS AQUI QUE ESTE ARGUMENTO ES MENOS VALIDO HOY CON LA DISPONIBILIDAD DE MUY LARGAS Y ECONOMICAS MEMORIAS.
- LOS LARGOS TAMANOS DE PAGINAS, GRAN CANTIDAD DE INFORMACION QUE ULTIMAMENTE REFERENCIADA ES PAGINADA DENTRO DE ALMACENAMIENTO PRIMARIO. ESTE PUNTO ES NECESARIO PARA PAGINAS PEQUENAS.
- PORQUE LA TRANSFERENCIA DE I/O DESDE DISCO ES RELATIVAMENTE TIEMPO-CONSUMIDO, DESEAMOS MINIMIZAR EL NUMERO DE TRANSFERENCIAS QUE UN PROGRAMA EXPERIMENTA EN LA CORRIDA. ESTE ES UN PUNTO NECESARIO PARA LAS PAGINAS LARGAS.
- VIGILAR PROGRAMAS PARA EXHIBIR LA PROPIEDAD DE LOCALIZACION DE REFERENCIA Y VIGILAR ESAS LOCALIDADES PARA SER REDUCIDAS. ASI UN PROGRMA CORTO AYUDARIA A UN PROGRAMA A ESTABLECER UN CONJUNTO DE TRABAJO FIRME.
- PORQUE EL FUNCIONAMIENTO Y LAS UNIDADES DE DATOS RARAMENTE ABARCAN UN NUMERO ENTERO DE PAGINAS, LOS SISTEMAS DE PAGINACION EXPERIMENTAN UNA FRAGMENTACION INTERNA. UN SEGMENTO DE LONGUITUD s ES COMO TENER SU ULTIMA PAGINA CASI LLENA COMO CASI VACIA, HASTA AQUI, HAY UNA MEDIA PAGINA DE FRAGMENTACION INTERNA (CON LA RESTRICCION DE QUE UNA PAGINA NO PUEDE CONTENER PARTICIONES DE MAS DE UN SEGMENTO). MENOR EL TAMANO DE LA PAGINA, MENOR LA FRAGMENTACION INTERNA.

LA MAYORIA DE LOS RESULTADOS EN LA LITERATURA TEORICOS Y EMPIRICOS APUNTAN LA NECESIDAD PARA PAGINAS PEQUENAS.

8.17 PROGRAMA DEL FUNCIONAMIENTO BAJO PAGINACION

PAGINACION ES UN CONCEPTO VALIOSO Y PROFUNDAMENTE INTEGRADO A LOS SISTEMAS DE COMPUTADORAS. MUCHOS ESTUDIOS HAN SIDO EJECUTADOS, EXAMINANDO EL FUNCIONAMIENTO DE PROCESOS EN AMBIENTES DE PAGINACION.

EL PROMEDIO DEL TIEMPO INTERFAULT, ES DECIR EL TIEMPO ENTRE FALLAS DE LA MAQUINA VARIA COMO EL NUMERO DE MARCO DE PAGINA PARA INCREMENTAR UN PROCESO. LA GRAFICA ES INCREMENTADA MONOTICAMENTE, LOS MARCOS DE PAGINA QUE UN PROCESO TIENE EL TIEMPO MAS LARGO ENTRE FALLAS DE PAGINA. PERO LA PAGINA SE ENCUENTRA EN UN PUNTO Y LO ASCENDENTE REDUCE LO PRONUNCIADO. ESTE ES EL PUNTO EN EL CUAL EL PROCESO TIENE SU CONJUNTO DE TRABAJO EN ALMACENAMIENTO PRIMARIO UNA VEZ QUE EL ALMACENAMIENTO PRIMARIO ES ASIGNADO SUFICIENTE PARA MANTENER EL CONJUNTO DE TRABAJO, SU FUNCION SE CURVA PRONUNCIADAMENTE INDICANDO QUE EL EFECTO DE ASIGNAMIENTO ADICIONAL DE LOS MARCOS DE PAGINA SOBRE EL INCREMENTO TIEMPO DE INTERFAULT NO ES MUY GRANDE. LA CLAVE ES OBTENER AL CONJUNTO DE TRABAJO DENTRO DEL ALMACENAMIENTO PRIMARIO.
UN EXPERIMENTO EN PARTICULAR ES CUANDO EL MAXIMO DE LA CURVA OCURRE CERCA DE 200 INSTRUCCIONES POR UN TAMANO DE PAGINA DE 1024 PALABRAS. RESULTADOS COMO ESTE VEMOS APUNTAN A LA NECESIDAD DE TAMANO DE PAGINA MENORES.

LAS DISCUSIONES CUALITATIVAS DE ESTA SECCION GENERALMENTE APUNTAN A LA VALIDACION DEL CONCEPTO DE CONJUNTO DE TRABAJO Y LA NECESIDAD DE TAMANOS DE PAGINA MENORES.

JOBS Y PROCESAMIENTO SCHEDULING

9.1 INTRODUCCION

LA ASIGNACION DE PROCESADORES FISICOS A PROCESADORES QUE PERMITAN EFECTUAR TRABAJOS A PROCESOS. LA ASIGNACION ES UN PROBLEMA COMPLEJO MANEJADO POR EL SISTEMA OPERATIVO. EN ESTAS SECCIONES DISCUTIREMOS LOS PROBLEMAS DE DETERMINAR CUANDO EL PROCESADOR DEBERA SER ASIGNADO Y A CUAL PROCESO. ESTO ES LLAMADO PROGRAMACION DE LOS PROCESADORES O PLANEACION DE LOS PROCESADORES.

9.2 NIVELES DE SEGUIMIENTOS

TRES NIVELES IMPORTANTES DE SEGUIMIENTOS SON CONSIDERADOS A CONTINUACION:

- SEGUIMIENTO DE ALTO NIVEL: ALGUNAS VECES LLAMADO INTINERARIO DE TRABAJO, ESTOS DETERMINAN CUALES JOBS SERAN ADMITIDOS A UNA ACTIVACION COMPLETA PARA LOS RECURSOS EXISTENTES. ESTO ES ALGUNAS VECES LLAMADO HORARIO DE ADMISION PORQUE DETERMINA CUALES JOBS GANARAN LA ENTRADA AL SISTEMA.

- SEGUIMIENTO DE NIVEL INTERMEDIO: ESTE DETERMINA CUAL PROCESO SERA ADMITIDO COMPLETO PARA EL CPU.

EL NIVEL INTERMEDIO RESPONDE A FLUCTUACIONES DE TERMINOS CORTOS EN LA CARGA DEL SISTEMA POR ACTIVADORES Y SUSPENCIONES TEMPORALES DEL PROCESO A REALIZAR FACILMENTE EL SISTEMA DE OPERACION Y REALIZA CIERTA AYUDA EN LA AMPLIACION DEL SISTEMA EJECUTABLE. HASTA AQUI EL SEGUIMIENTO DEL NIVEL INTERMEDIO ACTUA COMO UN BUFFER ENTRE LA ADMISION DE JOBS A EL SISTEMA Y LA ASIGNACION DEL CPU A ESTOS JOBS.

- SEGUIMIENTO DE BAJO NIVEL: ESTA DETERMINA CUALES PROCESOS ESTAN LISTOS PARA SER ASIGNADOS AL CPU, CUANDO ESTOS ESTEN DISPONIBLES Y ASIGNEN REALMENTE EL CPU A ESTE PROCESO (TERMINA O DESPACHA EL CPU AL PROCESO) EL SEGUIMIENTO DE BAJO NIVEL ES EJECUTADO POR EL DESPACHADOR QUE OPERA MUCHAS VECES POR EL SEGUNDO.

EL DESPACHADOR POR CONSIGUIENTE PERMANECE TODO EL TIEMPO DENTRO DEL ALMACENAMIENTO PRIMARIO.

9.3 OBJETIVOS DE LOS SEGUIMIENTOS

UNA DISIPLINA DE SEGUIMIENTOS DEBE SER:

- EQUITATIVA: LA DISIPLINA DE SEGUIMIENTOS ES EQUITATIVA SI TODOS LOS PROCESOS SON TRATADOS IGUALMENTE Y NO PODRAN ESTOS SUFRIR UN APLAZAMIENTO INDEFINIDO.

- MAXIMO TRABAJO POR UNIDAD DE TIEMPO (THROUGHPUT): LA DISIPLINA DE SEGUIMIENTO DEBE SER TENTADORA A SERVIR EL

MAYOR NUMERO POSIBLE DE PROCESOS POR UNIDAD DE TIEMPO.

- MAXIMIZAR: EL NUMERO DE USUARIOS ACTIVOS QUE RECIBAN UNA RESPUESTA EN UN TIEMPO ACEPTABLE (POCOS SEGUNDOS).

- SER PREDECIBLE: UN JOB DADO ESTARA CORRIENDO CON LA CONDICION DE DECIDIR CUALES PAGINAS EN ALMACENAMIENTO PRIMARIO SERAN DESPLAZADAS

- LLEVAR BALANCE ENTRE UTILIZACION Y RESPUESTA: EL MEJOR CAMINO QUE GARANTIZA UN BUEN TIEMPO DE RESPUESTA ES EL QUE TIENE SUFICIENTES RECURSOS DISPONIBLES SIEMPRE QUE ESTOS SEAN NECESITADOS. EL PRECIO HA SER PAGADO POR ESTA ESTRATEGIA QUE SOBRE TODO LA UTILIZACION DEBE SER POBRE. EN SISTEMAS EL TIEMPO REAL Y LA RAPIDA RESPUESTA ES ESENCIAL Y LA UTILIZACION DE RECURSOS ES MENOS IMPORTANTE. EN OTROS TIPOS DE SISTEMAS ECONOMICOS FRECUENTEMENTE HACIAN IMPERATIVA LA UTILIZACION EFECTIVA DE RECURSOS.

- EVITAR POSPOSICION INDEFINIDA: EN CUALQUIER CASO LA POSPOSICION INDEFINIDA PUEDE SER MALA COMO UN NUDO (DEADLOCK). EVITANDO LA POSPOSICION INDEFINIDA ES MEJOR CUMPLIDO POR " AGING" COMO UN PROCESO EN ESPERA DE UN RECURSO.

- LAS RESTRICCIONES DEL CPU DE UN PROCESO: CUANDO UN PROCESO ADQUIERE EL CPU ESTE VIGILA EL USO DEL CPU HASTA QUE TERMINA.

- YA SEA UN PROCESO BATCH O INTERACTIVO: GENERALMENTE LOS USUARIOS INTERACTIVOS SOLICITAN TRIVIALIDADES SOMETIDAS QUE DEBERAN RECIBIR SERVICIO INMEDIATO PARA GARANTIZAR UN BUEN TIEMPO DE RESPUESTA.
LOS USUARIOS BATCH NO ESTAN PRESENTES Y GENERALMENTE SUFREN DEMORAS JUSTAS.

- URGENCIA EN TIEMPO DE RESPUESTA: EN LA NOCHE UN PROCESO BATCH NO NECESITARA UNA RESPUESTA INMEDIATA SI UN PROCESO DE TIEMPO REAL DE UN SISTEMA DE CONTROL MONITOREA UNA GASOLINERA O REFINERIA, REQUIEREN UNA RAPIDA RESPUESTA PARA PREVENIR UNA POSIBLE EXPLOSION.

- PRIORIDAD DE PROCESOS: LOS PROCESOS DE ALTA PRIORIDAD DEBEN RECIBIR MEJOR TRATAMIENTO QUE AQUELLOS DE BAJA PRIORIDAD.

- CON QUE FRECUENCIA SE GENERAN PAGINAS DE ERROR: PRESUMIBLEMENTE LOS PROCESOS GENERAN POCAS PAGINAS DE ERROR TENIENDO ACUMULADOS ESTOS TRABAJOS DENTRO DEL ALMACENAMIENTO PRINCIPAL. EXPERIMENTANDO PROCESOS DE GRAN NUMERO DE PAGINAS DE ERROR ESTOS NO TIENEN TODAVIA ESTABLECIDO DENTRO DE SUS TRABAJOS INICIADOS. LA SABIDURIA CONVENCIONAL ES UN PROCESO FAVORECIDO QUE TIENE ESTABLECIDO DENTRO SU TRABAJO.

OTRO PUNTO DE VISTA ES QUE PROCESOS CON GRAN PORCENTAJE DE PAGINAS DE ERROR DEBE RECIBIR PRIORIDAD PORQUE ESTE USA BREVEMENTE EL CPU SOLAMENTE ANTES DE GENERAR UN REQUISITO DE I/O.

CON QUE FRECUENCIA ESTA SIENDO REMOVIDO POR UN PROCESO

DE ALTA PRIORIDAD FRECUENTEMENTE LOS PROCESOS REMOVIDOS DEBEN RECIBIR TRATAMIENTO MENOS FAVORECIDO. EL PUNTO ES QUE TODO EL TIEMPO EL S.O. INVIERTE EL OVERHEAD A OBTENER UN PROCESO CORRIENDO.

EL TIEMPO CORTO DE CORRIDA ANTES DE SER REMOVIDO NO JUSTIFICA LA OBTENCION DE OVERHEAD DEL PROCESO CORRIENDO EN PRIMER LUGAR.

- CUANTO TIEMPO REAL DE EJECUCION TIENE QUE RECIBIR EL PROCESO: ALGUNOS DISEMADORES SIENTEN QUE UN PROCESO QUE RECIBIO POCO TIEMPO DE EJECUCION DEBE SER FAVORECIDO.

OTROS CREEN QUE UN PROCESO QUE HA RECIBIDO MUCHO TIEMPO DE EJECUCION DEBE ESTAR CERCA DE SU TERMINACION Y DEBE SER FAVORECIDO CON UNA RICA COMPLEMENTACION Y DEJAR EL SISTEMA LO MAS PRONTO POSIBLE.

- CUANTO MAS NECESITA UN PROCESO PARA TERMINAR: LOS PROMEDIOS DE TIEMPO DE ESPERA PUEDEN SER MINIMIZADOS CORRIENDO PRIMERO LOS PROCESOS QUE REQUIEREN EL TIEMPO MINIMO HASTA SU TERMINACION. DESAFORTUNADAMENTE ESTO ES RARAMENTE UN CONOCIMIENTO EXACTO DE CUANTO TIEMPO MAS NECESITA CADA PROCESO POR SU TERMINACION.

9.4 ASIGNACION DE REMOCION VS NO REMOCION

LA DISIPLINA DE ASIGNACION ES NO REMOVIDA SI UN UNICO PROCESO ES DADO AL CPU Y ESTE NO PUEDE TOMAR OTRO CAMINO DESDE ESTE PROCESO. LA DISIPLINA DE ASIGNACION ES REMOVIBLE SI EL CPU PUEDE TOMAR OTRO CAMINO.

LA ASIGNACION REMOVIBLE ES USADA EN SISTEMAS EN EL CUAL PROCESOS DE ALTA PRIORIDAD SOLICITAN UNA RAPIDA ATENCION. EN SISTEMAS DE TIEMPO- REAL POR EJEMPLO, LAS CONSECUENCIAS DE PERDIDA POR UNA INTERRUPCION PUEDE SER ARRUINADA. DENTRO DE UN SISTEMA INTERACTIVO DE USO COMPARTIDO DE ASIGNACION REMOVIBLE ES IMPORTANTE SI NOS GARANTIZA UN TIEMPO DE RESPUESTA ACEPTABLE.

EL CONTEXTO INVOLUCRA UN SWITCHEO DE OVERHEAD ESTO HACE LA REMOCION, CUALQUIER PROCESO PUEDE SER GUARDADO DENTRO DEL ALMACENAMIENTO PRINCIPAL. HASTA QUE EL OTRO PROCESO SEA LEIDO NORMALMENTE POR EL CPU CUANDO SE SIENTA DISPONIBLE. TAMBIEN GUARDANDO PROGRAMAS QUE NO ESTAN CORRIENDO DENTRO DEL ALMACENAMIENTO PRINCIPAL, TAMBIEN INVOLUCRA OVERHEAD. EN LOS SISTEMAS NO REMOVIBLES, JOBS CORTOS SON HECHOS ESPERAR POR JOBS GRANDES, PERO EL TRATAMIENTO DE TODOS LOS PROCESOS ES IMPARCIAL. EL TIEMPO DE RESPUESTA ES MAS PREDECIBLE PORQUE SI LLEGAN JOBS DE MAS ALTA PRIORIDAD ESTOS NO PUEDEN DESPLAZAR A LOS JOBS QUE ESTAN EN ESPERA. DISENANDO UN MECANISMO DE ASIGNACION REMOVIBLE, UNO DEBE CUIDADOSAMENTE CONSIDERAR LAS ARBITRARIEDADES VIRTUALMENTE DE CUALQUIER ESQUEMA DE PRIORIDADES.

NOSOTROS ESTRUCTURAREMOS UN MECANISMO SOFISTICADO A IMPLEMENTAR UN ESQUEMA FIELMENTE DE PRIORIDADES REMOVIBLES CUANDO EN REALIDAD LAS PRIORIDADES MISMAS NO SON ASIGNADAS SIGNIFICATIVAMENTE. ESTO ES COMUN DENTRO DE LOS SISTEMAS OPERATIVOS TENER MECANISMOS ILUSORIOS (FANTASIOSOS) SOPORTANDO ALGUNOS ESQUEMAS ARBITRARIOS. EL DISENADOR DEBE SER SENSATO PARA EVALUAR CADA MECANISMO PROPUESTO CUIDADOSAMENTE ANTES DE IMPLANTARLO "MANTENERLO SIMPLE", TIENE GRAN ATRACTIVO, PERO SI NO PODEMOS MANTENERLO SIMPLE NOSOTROS DEBEMOS AL MENOS INSISTIR EN HACERLO EFECTIVO Y SIGNIFICATIVO.

9.5 RELOJ DE INTERRUPCIONES

EL PROCESO AL CUAL EL CPU ESTA ASIGNADO ACTUALMENTE SE DICE QUE ESTA CORRIENDO, SI ES UN PROCESO DEL SISTEMA OPERATIVO ENTONCES EL SISTEMA OPERATIVO ESTA CORRIENDO Y PUEDE HACER DESICIONES QUE INFLUYAN EN LA DESICION DEL SISTEMA. PARA PREVENIR QUE LOS USUARIOS MONOPOLIZEN EL SISTEMA (YA SEA MALICIOSA O ACCIDENTALMENTE) EL SISTEMA OPERATIVO TIENE MECANISMOS PARA QUITAR EL CPU AL USUARIO. EL SISTEMA OPERATIVO INICIALIZA EL RELOJ DE INTERRUPCIONES O RELOJ DE INTERVALOS DE TIEMPO PARA GENERAR UNA INTERRUPCION A UN TIEMPO FUTURO ESPECIFICO (O EN ALGUN LAPSO DE TIEMPO EN EL FUTURO). EL CPU ES ENTONCES DESPACHADO AL SIGUIENTE PROCESO. EL PROCESO RETIENE EL CONTROL DEL CPU HASTA QUE LO LIBERA VOLUNTARIAMENTE O EL RELOJ INTERRUMPE O ALGUNA OTRA INTERRUPCION DESVIA LA ATENCION DEL CPU, SI EL PROCESO DE UN USUARIO ESTA CORRIENDO Y UNA INTERRUPCION DE RELOJ OCURRE ESTA CAUSA QUE EL SISTEMA OPERATIVO CORRA O SE EJECUTE. EL SISTEMA OPERATIVO ENTONCES DECIDE QUE PROCESO OBTENDRA EL CPU EN SEGUIDA. EL RELOJ DE INTERRUPCION AYUDA A GARANTIZAR TIEMPO DE RESPUESTA RAZONABLE PARA USUARIOS INTERACTIVOS, PREVIENE AL SISTEMA QUEDAR COLGADO A UN USUARIO EN UN CICLO INFINITO Y PERMITE A LOS PROCESOS RESPONDER A EVENTOS DEPENDIENTES DE TIEMPO. PROCESOS QUE NECESITAN CORRER PERIODICAMENTE DEPENDEN DEL RELOJ DE INTERRUPCIONES.

9.6 PRIORIDADES

LAS PRIORIDADES PUEDEN SER ASIGNADAS AUTOMATICAMENTE POR EL SISTEMA ES DECIR ASIGNADAS EXTERNAMENTE. PUEDEN SER MERECIDAS O COMPARADAS. PUEDEN SER RACIONALMENTE ASIGNADAS O PUEDEN SER ARBITRARIAMENTE ASIGNADAS EN SITUACIONES EN LAS CUALES UN MECANISMO DEL SISTEMA NECESITA DISTINGUIR ENTRE PROCESO PERO REALMENTE CUIDA CUAL ES VERDADERAMENTE IMPORTANTE.

9.7 PRIORIDADES ESTATICAS VS PRIORIDADES DINAMICAS

LAS PRIORIDADES ESTATICAS NO CAMBIAN. LOS MECANISMOS DE PRIORIDADES ESTATICAS SON FACILES DE IMPLEMENTAR Y TIENEN RELATIVAMENTE BAJO OVERHEAD. NO SON SIN EMBARGO SUCEPTIBLES A CAMBIOS EN EL AMBIENTE; CAMBIOS QUE PODRIAN HACER DESEABLES AJUSTAR UNA PRIORIDAD. LOS MECANISMOS DE PRIORIDAD DINAMICA: RESPONDEN A CAMBIOS. LA PRIORIDAD INICIAL ASIGNADA A UN PROCESO PUEDE TENER SOLO UNA CORTA DURACION DESPUES DE LA CUAL ES AJUSTADA A UN MEJOR VALOR. ESQUEMAS DE PRIORIDAD MECANICA SON MAS COMPLEJAS A IMPLEMENTAR Y TIENEN GRAN OVERHEAD QUE LOS ESQUEMAS ESTATICOS. EL OVERHEAD ES UNA ESPERANZA JUSTIFICADA POR LA RESPUESTA DEL SISTEMA.

9.8 PRIORIDADES PURCHASEDT

UN BUEN SISTEMA OPERATIVO PROVEE SERVICIO COMPETENTE Y RAZONABLE A UNA GRAN COMUNIDAD DE USUARIOS PERO TAMBIEN DEBE DE PROVEER EN SITUACIONES EN LAS CUALES UN MIEMBRO DE LA COMUNIDAD DE USUARIOS NECESITA UN TRATAMIENTO ESPECIAL. UN USUARIO CON UN JOB APRESURADO DEBERA DE ESTAR DISPUESTO A PAGAR UN PRECIO POR UN SERVICIO DE ALTO NIVEL. ESTE CAMBIO EXTRA ES AMERITADO PORQUE LOS RECURSOS SERAN NECESITADOS O SER RETIRADAS DESDE OTRO CONSUMIDOR SI ESTOS NO DESEAN UN CAMBIO EXTRA ENTONCES TODOS LOS USUARIOS DEBERAN SOLICITAR O REQUERIR UN SERVICIO DE ALTO NIVEL.

9.9 ASIGNACION DE LINEA MUERTA (DEADLINE)

EN LA ASIGNACION DEADLINE CIERTOS JOBS SON ASIGNADOS PARA SER TERMINADOS POR UN TIEMPO ESPECIFICO O DEADLINE. ESTOS JOBS PUEDEN TENER VALORES ALTOS DELIBERADOS SOBRE EL TIEMPO Y PUEDEN ESTAR SIN VALOR SI LOS DELIBERAMOS DESPUES DE DEADLINE. EL USUARIO ESTA CON FRECUENCIA DISPUESTO A PAGAR UN PRECIO POR TENER EL SISTEMA EN TIEMPO COMPLETO. LA ASIGNACION DEADLINE ES COMPLEJA POR CUALQUIERA DE ESTAS RAZONES:

- EL USUARIO DEBERA SUPLIR LOS REQUERIMIENTOS PRECISOS SOLICITADOS DE LOS JOBS EN AVANCES, TAL COMO LA INFORMACION ES RARAMENTE DISPONIBLE.
- EL SISTEMA DEBERA CORRER EL DEADLINE JOBS SIN QUE EL SERVICIO SEA SEVERAMENTE DEGRADADO A OTRO USUARIO.
- EL SISTEMA DEBERA PROPONERSE CUIDADOSAMENTE ATRAVES DE LOS RECURSOS REQUERIDOS ATRAVES DEL DEADLINE. ESTO DEBE SER COMPLICADO PORQUE LOS JOBS NUEVOS DEBERAN LLEGAR Y DEMANDAMOS UN LUGAR IMPRESINDIBLE SOBRE EL SISTEMA.
- EN CUALQUIER JOB DEADLINE ESTARA UNICAMENTE ACTIVA LA ASIGNACION Y PODRA SER PUESTA TAN COMPLEJA QUE EL METODO DE OPTIMIZACION SOFISTICADA DEBERAN SER NECESITADOS TO ENSURE QUE EL DEADLINE ESTA ENCONTRANDO.
- LA INTENSIVIDAD ADMINISTRATIVA DE UN RECURSO

REQUERIDO POR LA ASIGNACION DEADLINE PUEDE GENERALMENTE COMO UN SUBSTANCIAL OVERHEAD. AUN CUANDO EL USUARIO ESTE DISPUESTO A PAGAR HONORARIOS SUFICIENTEMENTE ALTOS POR RECIBIR EL SERVICIO; EL CONSUMO DE LOS RECURSOS DEL SISTEMA DEBERA SER ALTA YA QUE EL RESTO DE LA COMUNIDAD DE USUARIOS PODRA SUFRIR UN SERVICIO DEGRADABLE.

TALES CONFLICTOS DEBEN SER CONSIDERADOS CUIDADOSAMENTE POR LOS DISENADORES DE LOS SISTEMAS OPERATIVOS.

9.10 ASIGNACION PRIMERO EN ENTRAR PRIMERO EN SALIR (FIRST-IN-FIRST-OUT (FIFO))

LOS PROCESOS SON DESPACHADOS DE ACUERO AL TIEMPO DE LLEGADAS SOBRE LA LISTA EN LA COLA (QUEVE) SOLO UN PROCESO TIENE AL CPU HASTA FINALIZARLO, FIFO ES UNA DISIPLINA SIN REMOCION O NO REMOVIBLE. ESTO ES JUSTO EN EL SENTIDO FORMAL PERO ALGO INJUSTO EN EL TIEMPO DE ESPERA HECHO POR LOS JOBS CORTOS Y LOS JOBS LARGOS Y LOS JOBS SIN IMPORTANCIA HACEN LA ESPERA COMO UN JOB CON IMPORTANCIA. FIFO OFRECE UNA VARIANZA RELATIVAMENTE PEQUENA DENTRO DE LOS TIEMPOS DE RESPUESTA Y ES POR ESO MAS PREDECIBLE QUE ALGUNOS OTROS ESQUEMAS, ESTO NO ES USUABLE DENTRO DE LOS USUARIOS DE ASIGNACION INTERACTIVA PORQUE NO PUEDE GARANTIZAR BUENOS TIEMPOS DE RESPUESTA.

FIFO ES USADA RARAMENTE COMO UN ESQUEMA MAESTRO EN LOS SISTEMAS DE HOY EN DIA PERO ESTOS ESTAN FRECUENTEMENTE DENTRO DE OTROS ESQUEMAS POR EJEMPLO, CUALQUIER ESQUEMA DE ASIGNACION DESPACHA A LOS PROCESOS DE ACUERDO A LA PRIORIDAD PERO PROCESOS CON LA MISMA PRIORIDAD SON DESPACHADOS POR FIFO.

9.11 ASIGNACION ROUND-ROBIN (RR)

EN LA ASIGNACION ROUND ROBIN LOS PROCESOS SON DESPACHADOS POR FIFO PERO ESTAN DADOS CON UNA CANTIDAD LIMITADA DE TIEMPO DEL CPU LLAMADA "TIME SLICE" O UN QUANTUM (TIEMPO PARTICIONADO O CATIDAD).

ESTO ES SI UN PROCESO NO TERMINA ANTES DE QUE TERMINE EL TIEMPO DEL CPU, ESTE SERA REMOVIDO AL FINAL DE LA LISTA DE ESPERA Y EL CPU SERA REMOVIDO AL PROXIMO PROCESO QUE ESPERA EN LA ROUND ROBIN, ES UNA ASIGNACION EFECTIVA DENTRO DEL AMBIENTE DE USUARIOS DE TIEMPO COMPARTIDO EN LA CUAL LOS SISTEMAS NECESITAN LA GARANTIA DE UN TIEMPO RAZONABLE DE RESPUESTA PARA USUARIOS INTERACTIVOS. LA REMOCION OVERHEAD ES KEPT DOW POR CONTEO EFICIENTE DE UN MECANISMO DE SWITCHEO POR PROVEER UN ADECUADO ALMACENAMIENTO PARA LOS PROCESOS QUE RECIDEN DENTRO DEL ALMACEN PRINCIPAL AL MISMO TIEMPO.

9.12 TAMANO DE QUANTUM

LA DETERMINACION DEL TAMANO DE QUANTUM ES CRITICA PARA LA OPERACION EFECTIVA DEL SISTEMA DE COMPUTADORAS DEBERIA SER EL QUANTUM GRANDE O PEQUENO? FIJO O VARIABLE? DEBERIA SER EL MISMO QUANTUM PARA TODOS LOS USUARIOS? O DEBERIA SER DETERMINADA SEPARADAMENTE POR EL USUARIO?. PRIMERO VAMOS A

SEGUN EL PROCESO SEA GRANDE A CADA PROCESO SE LE DA TANTO TIEMPO COMO ESTE NECESITE PARA COMPLETAR POR TANTO EL ESQUEMA ROUND ROBIN DEGENERA EN FIFO. SEGUN EL QUANTUM SEA MUY CORTO, EL CONTENIDO CAMBIANDO ENCIMA VIENE A SER UN FACTOR DOMINANTE Y EL RENDIMIENTO DEL SISTEMA EVENTUALMENTE DE GRADO AL PUNTO QUE LA MAYORIA DEL TIEMPO ES GASTADO CAMBIANDO AL PROCESADOR, UN POCO SI ACASO, TIEMPO GASTADO EN LA EJECUCION DE LOS CALCULOS DE LOS USUARIOS.

DONDE JUSTAMENTE ENTRE CERO E INFINITO DEBERIA SER PUESTO EL QUANTUM. CONSIDERE EL SIGUIENTE EXPERIMENTO. SUPONGA QUE HAY UN DISCO CIRCULAR MARCADO EN LECTURAS MARCADO DESDE Q=O HASTA Q=INFINITO. EMPEZAMOS CON EL BOTON POSICIONADO EN O. SEGUN EL BOTON ES GIRADO EL QUANTUM PARA EL SISTEMA CAMBIA ASUME QUE EL SISTEMA ES OPERACIONAL Y QUE HAY MUCHOS USUARIOS INTERACTIVOS. SEGUN EL BOTON ES INICIALMENTE GIRADO, LAS LECTURAS ESTAN CERCA DE O Y EL CONTENIDO CAMBIADO ENCIMA CONSUME LA MAYORIA DE LOS RECURSOS DEL CPU. LOS USUARIOS INTERACTIVOS EXPERIMENTAN UN SISTEMA PEREZOSO CON TIEMPOS DE RESPUESTA POBRES. SEGUN EL BOTON ES GIRADO MAS ALLA INCREMENTANDO EL QUANTUM, LOS TIEMPOS DE RESPUESTA MEJORAN AL MENOS EL PUNTO HA SIDO ALCANZADO EN EL CUAL EL PORCENTAJE CONSUMIENDO POR OVERHEAD ES LO BASTANTE PEQUENO QUE LOS USUARIOS RECIBEN ALGUN SERVICIO DEL CPU, PERO LOS TIEMPOS DE RESPUESTA AUN NO SON MUY BUENOS.

-SEGUN EL BOTON ES GIRADO MAS, LOS TIEMPOS DE RESPUESTA CONTINUAN MEJORANDO EN UN PUNTO, LOS USUARIOS ESTAN OBTENIENDO RESPUESTAS PUNTUALES DESDE EL SISTEMA. PERO AUN NO ESTAN CLAROS EL QUANTUM PUESTO ES OPTIMO. EL BOTON ES GIRADO UN POCO MAS ALLA Y LOS TIEMPOS DE RESPUESTA VIENEN A SER LIGERAMENTE MEJORES. PERO LUEGO, SEGUN EL BOTON ES GIRADO MAS, LOS TIEMPOS DE RESPUESTA VIENEN A SER LENTOS OTRA VEZ. SEGUN EL QUANTUM SE HACE MAS GRANDE PARA EL USUARIO CORRER COMPLETO RECIBIENDO AL CPU. EL SCHEDULING ESTA DEGENERADO EN FIFO EN EL CUAL LOS PROCESOS MAS LARGOS HACEN QUE LOS MAS CORTOS ESPEREN Y EL TIEMPO ESPERADO PROMEDIO SE INCREMENTA SEGUN LOS PROCESOS MAS LARGOS CORRAN COMPLETOS ANTES DE DEJAR AL CPU.

- CONSIDERE EL SUPUESTO VALOR OPTIMO DEL QUANTUM QUE CEDIO BUENOS TIEMPOS DE RESPUESTA. ESTOS SON UNA PEQUENA FRACCION DE UN SEGUNDO. QUE REPRESENTA ESE QUANTUM ? ES BASTANTE GRANDE DE TAL MANERA QUE LA MAYORIA DE LAS PREGUNTAS INTERACTIVAS REQUIEREN MENOR TIEMPO QUE LA DURACION DEL QUANTUM. CUANDO UN PROCESO INTERACTIVO EMPIEZA A EJECUTARCE NORMALMENTE USA EL CPU LO SUFICIENTEMENTE

GRANDE PARA GENERAR UNA PREGUNTA DE I/O, UNA VEZ QUE I/O Es GENERADO ESE PROCESO CEDE EL CPU AL SIGUIENTE PROCESO PORQUE EL QUANTUM ES MAS GRANDE QUE ESTE TIEMPO DE CALCULO HASTA I/O, LOS PROCESOS DE USUARIO SON GUARDADOS TRABAJANDO A VELOCIDAD SUPERIOR CADA VEZ QUE UN PROCESO DE USUARIO CONSIGUE AL CPU HAY GRAN PROBABILIDAD DE QUE CORRA HASTA QUE GENERA UN I/O, ESTE MINIMIZA EL OVERHEAD OCUPADO, MAXIMIZA LA UTILIZACION DE I/O Y PROVEE RELATIVAMENTE RAPIDOS TIEMPOS DE RESPUESTA.

-CUAL ES EL QUANTUM OPTIMO ? CLARAMENTE ESTO VARIA DE UN SISTEMA A OTRO Y VARIA BAJO DIFERENTES CARGOS. ESTO TAMBIEN VARIA DE PROCESO A PROCESO, PERO NUESTRO EXPERIMENTO NO ESTA EQUIPADO PARA MEDIR DIFERENCIAS EN PROCESOS.

- CUANDO TODOS LOS PROCESOS ESTAN EN EL LIMITE DEL CPU, ESTE HACE POCO SENTIDO A PROCESOS OCUPADOS. EL PUNTO ES QUE EL OVERHEAD ADICIONAL ENVUELTO EN PREEMPTION SOLO QUITA DESDE EL RENDIMIENTO DEL SISTEMA TAN LARGO COMO ALGUNOS USUARIOS INTERACTIVOS ESTEN EN LA MEZCLA DE MULTIPROGRAMACION, SIN EMBARGO EL CPU DEBERA ESTAR AUN OCUPADO PERIODICAMENTE PARA GARANTIZAR TIEMPO DE RESPUESTA INTERACTIVOS.

9.13 SCHEDULING SJF

-SJF (PRIMER JOB MAS CORTO) ES UNA DISIPLINA SCHEDULING NO-PREVACIADA EN LA CUAL EL JOB ESPERANDO O PROCESO CON EL TIEMPO DE CORRIDA COMPLETO ESTIMADO FEQUEMO ES EL SIGUIENTE QUE CORRE SJF REDUCE EL TIEMFO ESFERADO PROMEDIO SOBRE FIFO. LOS TIEMPOS ESPERANDO, SIN EMBARGO TIENEN UNA VARIANZA MAS GRANDE, ESTO ES SON MAS IMPREDECIBLES QUE FIFO, ESPECIALMENTE PARA JOBS GRANDES.

-SJF FAVORECE A LOS JOBS CORTOS (O PROCESOS) A EXPENSAS DE OTROS MAS LARGOS MUCHOS DISENADORES DEFIENDEN QUE EL MAS CORTO EL JOB, EL MEJOR SERVICIO DEBIERA RECIBIR. NO HAY UN ACUERDO UNIVERSAL EN ESTO, ESPECIALMENTE CUANDO LAS PRIORIDADES DEL JOB DEBEN SER CONSIDERADAS.

-SJF SOLUCIONA JOBS PARA SERVICIO EN UNA MANERA QUE SE ASEGURA QUE EL SIGUIENTE JOB COMPLETARA Y DEJARA EL SISTEMA TAN PRONTO COMO SEA POSIBLE. ESTO TIENDE A REDUCIR EL NUMERO DE JOBS ESPERADO Y TAMBIEN REDUCE EL NUMERO DE JOBS ESPERADOS DETRAS DE LOS JOBS GRANDES COMO RESULTADO, SJF PUEDE MINIMIZAR EL TIEMPO ESPERADO PROMEDIO DE JOB SEGUN PASAN A TRAVES DEL SISTEMA.

-EL PROBLEMA OBVIO CON SJF ES QUE REQUIERE CONOCIMIENTOS PRECISOS DE QUE TAN LARGO UN JOB O PROCESO CORRERA Y ESTA INFORMACION NO ESTA USUALMENTE DISPONIBLE. LO MEJOR QUE SJF PUEDE HACER ES CONTAR CON ESTIMADORES USUARIOS DE TIEMPO DE CORRIDA. EN AMBIENTES DE PRODUCCION DONDE LOS MISMOS JOBS CORREN REGULARMENTE, PUEDE SER POSIBLE PROVEER

ESTIMACIONES RAZONABLES. PERO EN AMBIENTES DE DESARROLLO LOS USUARIOS SABEN QUE TAN LARGOS SON LOS PROGRAMAS QUE SE EJECUTARAN.

EL CONFIAR EN ESTIMACIONES DE USUARIOS TIENE UNA INTERESANTE RAMIFICACION. SI LOS USUARIOS SABEN QUE EL SISTEMA ESTA DISENADO PARA FAVORECER JOBS CON PEQUENOS TIEMPOS DE CORRIDA ESTIMADOS ELLOS PUEDEN DAR ESTIMACIONES PEQUENAS, EL SCHEDULER PUEDE SER DISENADO, SIN EMBARGO PARA REMOVER ESTA TENTACION EL USUARIO PUEDE SER AVISADO DE QUE SI EL JOB CORRE MAS LARGO QUE LO ESTIMADO ESTE SERA TERMINADO Y EL USUARIO SERA CARGADO POR EL TRABAJO. UNA SEGUNDA OPCION ES CORRER EL JOB EN EL TIEMPO ESTIMADO MAS UN PEQUENO PORCENTAJE EXTRA Y LUEGO ACTIVARLO, ESTO ES PRESERVARLO EN SU FORMA ACTUAL DE TAL MANERA QUE PUEDA SER REESTABLECIDO EN UN TIEMPO MAS TARDE. EL USUARIO POR SUPUESTO, PAGARIA POR ARCHIVARLO Y DESARCHIVARLO (REESTABLECERLO) DE ENCIMA Y SUFRIRIA UNA DEMORA EN LA TERMINACION DEL JOB. OTRA SOLUCION ES CORRER EL JOB EN EL TIEMPO ESTIMADO A PROMEDIOS NORMALES DE ENVIO Y LUEGO CARGA UN PROMEDIO PRIMO, BIEN ARRIBA DE LOS CARGOS NORMALES PARA TIEMPOS DE EJECUCION ADICIONAL. BAJO ESTE ARREGLO, EL USUARIO PROVEE IRREALISTICAMENTE TIEMPO DE CORRIDA ESTIMADOS BAJOS PARA OBTENER MEJOR SERVICIO PAGARA ULTIMAMENTE UNA PRIMA AYUDA.

- SJF COMO FIFO ES NO PREVACIADA Y POR LO TANTO NO ES UTIL EN AMBIENTES DE TIEMPO COMPARTIO EN LOS CUALES TIEMPOS DE RESPUESTA RAZONABLES DEBEN SER GARANTIZADOS.

9.14 SCHEDULING TIEMPO-RESTANTE-MAS CORTO (SRT)

- SRT ES LA CONTRAPARTE PREVACIADA DE SIF Y ES UTIL EN TIEMPO COMPARTIDO. EN SRT EL PROCESO CON EL TIEMPO DE CORRIDA ESTIMADO MAS CORTO PARA TERMINAR CORRE ENSEGUIDA INCLUYENDO NUEVAS LLEGADAS. EN SJF UNA VEZ QUE UN JOB INICIA A EJECUTARSE, ESTE CORRE HASTA TERMINAR. EN SRT UN PROCESO CORRIENDO PUEDE SER OCUPADO POR UN NUEVO PROCESO CON UN TIEMPO DE CORRIDA ESTIMADO MAS CORTO. OTRA VEZ SRT REQUIERE ESTIMACIONES DEL FUTURO PARA SER EFECTIVO Y EL DISENADOR DEBE PROVEER PARA USUARIOS POTENCIALES EL ABUSO DE ESTRATEGIAS SCHEDULING DEL SISTEMA.

- SRT TIENE MAS ALTO OVERHEAD QUE SJF, ESTE DEBE GUARDAR EL ROSTRO DEL TIEMPO DE SERVICIO TRANSCURRIDO QUE SEAN REGISTRADOS Y ESTO CONTRIBUYE AL OVERHEAD DEL ESQUEMA. TEORICAMENTE SRT OFRECE TIEMPOS DE ESPERA MINIMOS. PERO DEBIDO AL OVERHEAD ES POSBLE QUE EL SJF PUEDA ACTUALMENTE EJECUTAR MEJOR EN CIERTAS SITUACIONES.

- SUPONGA QUE UN JOB CORRIENDO ESTA CASI COMPLETO Y LUEGO UN NUEVO JOB CON UN TIEMPO DE SERVICIO ESTIMADO MUY PEQUENO. DEBERIA ESTAR EL JOB QUE ESTA CORRIENDO OCUPADO ?

LA DISIPLINA SRT PARA EJECUTAR LA OCUPACION PERO REALMENTE VALE LA PENA ? ESTA ITUICION PUEDE SER MANEJADA CONSTRUYENDO UN VALOR DE ENTRADA DE TAL MANERA QUE UNA VEZ QUE UN JOB ESTA CORRIENDO NECESITE MAS QUE ESTA CANTIDAD DE TIEMPO, PARA COMPLETAR EL SISTEMA GARANTIZA QUE ESTE CORRERA COMPLETO SIN SER INTERRUMPIDO.

- SUPONGA QUE LLEGA UN JOB CUYO TIEMPO DE SERVICIO ESTIMADO ES SOLO LIGERAMENTE MENOS QUE EL RESTANTE PARA UN JOB COMIENDO CON MUCHO PROCESAMIENTO RESTANTE. AQUI TAMBIEN EL SRT PURO EJECUTARA LA OCUPACION PERO SI LA OCUPACION OVERHEAD ES MAYOR QUE LA DIFERENCIA EN LOS TIEMPOS DE SERVICIO DE LOS 2 JOBS, OCUPANDO EL JOB CORRIENDO ACTUALMENTE RESULTA DE MAS POBRE RENDIMIENTO. EL PUNTO DE TODO ESTO QUE LOS DISENADORES DE SISTEMAS OPERATIVOS DEBEN VER CUIDADOSAMENTE ES EL OVERHEAD DE LOS MECANISMOS DE ADMISION DE LOS RECURSOS CONTRA LOS BENEFICIOS ANTICIPADOS.

9.15 SIGUIENTE PROPORCION DE MAS ALTA RESPUESTA (SCHEDULING HRN HIGHEST-RESPONSE-RATIO-NEXT)

- BRINCH HANSEN DESARROLLO LA ESTRATEGIA HRN QUE CORRIGUE ALGUNOS DE LOS DEFECTOS EN SJF, PARTICULARMENTE LA EXCESIVA INCLINACION CONTRA LOS JOBS MAS LARGOS Y EL EXCESIVO FAVORITISMO HACIA NUEVOS JOBS CORTOS. HRN ES UNA DISIPLINA SCHEDULING NO-PREVACIADA EN LA CUAL LA PRIORIDAD DE CADA JOB ES UNA FUNCION NO SOLO DEL TIEMPO DE SERVICIO DEL JOB SINO TAMBIEN DE LA CANTIDAD DE TIEMPO QUE EL JOB HA ESTADO ESPERANDO PARA EL SERVICIO. UNA VEZ QUE UN JOB OBTIENE EL CPU, ESTE CORRE HASTA TERMINAR LAS PRIORIDADES DINAMICAS EN HRN CALCULADAS DE ACUERDO A LA FORMULA:

$$\text{PRIORIDAD} = \frac{\text{TIEMPO DE ESPERA} + \text{TIEMPO DE SERVICIO}}{\text{TIEMPO DE SERVICIO}}$$

9.16 COLAS DE RETROALIMENTACION DE MULTINIVELES

- UN PROCESO OBTIENE AL CPU ESPECIALMENTE CUANDO ESTE NO TIENE TODAVIA LA SUERTE DE ESTABLECER UN PATRON DE CONDUCTA, EL SCHEDULER NO TIENE IDEA DE LA CANTIDAD PRECISA DE TIEMPO DE CPU QUE EL PROCESO NECESITARA. LOS PROCESOS DE LIMITE DE I/O PODRIAN USAR EL CPU SOLO EN BREVE ANTES DE QUE SE GENERE UNA PREGUNTA DE I/O.

-UN MECANISMO DE SCHEDULING DEBERIA FAVORECER JOBS CORTOS, FAVORECER JOBS EN EL LIMITE DE I/O PARA OBTENER BUENA UTILIZACION DE LOS DISPOSITIVOS DE I/O. DETERMINAR LA NATURALEZA DE UN JOB TAN RAPIDO COMO SEA POSIBLE Y EN CONSECUENCIA ASIGNA EL JOB.

- LAS COLAS DE RETROALIMENTACION DE MULTINIVELES PROVEEN UNA ESTRUCTURA QUE REALIZA ESTAS METAS. UN NUEVO

PROCESO ENTRA EN LA RED DE COLAS ATRAS DE LA COLA DE MAS ARRIBA. ESTA SE MUEVE ATRAVES DE ESA COLA FIFO HASTA QUE CONSIGUE EL CPU. SI EL JOB TERMINA O ABANDONA EL CPU PARA ESPERAR POR I/O A TERMINAR ALGUN EVENTO, EL JOB DEJA LA RED DE COLAS. SI EL QUANTUM EXPIRA ANTES DE QUE EL PROCESO ABANDONE VOLUNTARIAMENTE EL CPU, EL PROCESO ES COLOCADO EN LA PARTE DE ATRAS DE LA COLA DEL NIVEL MAS BAJO SIGUIENTE. EL PROCESO ES LUEGO SERVIDO CUANDO ESTE ALCANZA LA CABEZA DE ESA COLA SI LA PRIMER COLA ESTA VACIA. TAN LARGO COMO EL PROCESO CONTINUE USANDO EL QUANTUM PROVEIDO EN CADA NIVEL, ESTA CONTINUA METIENDOSE A LA PARTE DE ATRAS DE LA SIGUIENTE COLA MAS BAJA.

USUALMENTE HAY ALGUNA COLA DEL NIVEL BAJO ATRAVES DE CUAL PROCESO CIRCULA ROUND ROBIN HASTA QUE TERMINE.

- EN MUCHOS ESQUEMAS DE RETROALIMENTACION DE MULTINIVEL EL QUANTUM DADO AL PROCESO CUANDO SE MUEVE A CADA COLA DE NIVEL MAS BAJO VIENE A SER MAS GRANDE. ASI LO MAS LARGO QUE UN PROCESO HA ESTADO EN LA RED DE COLAS, ES EL QUANTUM GRANDE QUE LE ES ASIGNADO CADA VEZ QUE OBTIENE AL CPU. PERO NO PUEDE OBTENER EL CPU MUY FRECUENTEMENTE PORQUE LOS PROCESOS EN LAS COLAS MAS ALTAS SE LES HA ASIGNADO PRIORIDAD MAS ALTA. UN PROCESO EN UNA COLA DADA NO PUEDE CORRER A MENOS QUE TODAS LAS COLAS DE LOS NIVELES MAS ALTOS ESTEN VACIAS. UN PROCESO CORRIENDO ES OCUPADO POR UN PROCESO LLEGANDO EN UNA COLA MAS ALTA.

- AHORA QUE LA OPERACION DE LA RED DE COLAS HA SIDO DESCRITA, CONSIDERE COMO TAL MECANISMO RESPONDE A DIFERENTES TIPOS DE PROCESOS.

EL MECANISMO FAVORECERIA A LOS PROCESOS EN LOS LIMITES DE I/O A CONSEGUIR BUENA UTILIZACION DE DISPOSITIVOS Y RESPUESTA A USUARIOS INTERACTIVOS. VERDADERAMENTE DEBIDO A LOS PROCESOS EN EL LIMITE DE I/O ENTRARA LA RED CON MUY ALTA PRIORIDAD Y SERA DADA AL CPU RAPIDAMENTE. EL QUANTUM EN LA PRIMERA COLA ES ESCOGIDO PARA SER BASTANTE GRANDE DE TAL MANERA QUE LA GRAN MAYORIA DE LOS JOBS EN EL LIMITE DE I/O EMITIRAN UNA PREGUNTA DE I/O ANTES DE QUE EL PRIMER QUANTUM EXPIRE. CUANDO EL PROCESO PREGUNTE POR I/O ESTE DEJA LA RED Y VERDADERAMENTE TRATAMIENTO FAVORABLE HA RECIBIDO.

- AHORA CONSIDEREMOS UN JOB EN EL LIMITE DEL CPU QUE NECESITE UNA CANTIDAD GRANDE DE TIEMPO DE CPU. ESTE ENTRA A LA RED A LA COLA MAS ALTA CON MUY ALTA PRIORIDAD. ESTE RECIBE SU PRIMER DISPARO EN EL CPU RAPIDAMENTE, PERO SU QUANTUM EXPIRA Y EL PROCESO ES MOVIDO.