Cátedra de Sistemas operativos UTN - FRBA	
Título de documento	SHREK – Especificación de Trabajo práctico.
Contenido	El presente documento contiene la especificación formal del trabajo práctico número 3

Trabajo práctico Nro. 3

Cátedra de sistemas operativos

S.H.R.E.K.

(Scheduler Hogareño Residente En el Kernel)

Versión 1.9

Índice

Introducción

El trabajo práctico Nº 3 consiste en el desarrollo de una aplicación lenguaje C sobre Linux. La estructura del mismo consta de las siguientes dos partes:
La primera de ellas, presenta diversos consejos y ejercicios de baja complejidad orientados a ofrecer un primer acercamiento del alumno al lenguaje y a las herramientas que se usarán en el mismo.
La segunda, consiste en el desarrollo de una aplicación de mayor complejidad que atacará problemas específicos de diseño de sistemas operativos.

Objetivos del trabajo práctico

Que el alumno:

- Adquiera conceptos prácticos del uso distintas herramientas de programación (API) que brindan los sistemas operativos modernos.
- Entender algunos aspectos del diseño de un sistema operativo moderno.
- Afirmar diversos conceptos teóricos de un sistema operativo mediante la implementación práctica de algunos de ellos.
- Familiarizarse con técnicas de programación de sistemas, como por ejemplo el empleo de makefiles, archivos de configuración y archivos de log.
- Que los alumnos conozcan con grado de detalle la operatoria de Linux mediante la utilización de un lenguaje de programación de relativo bajo nivel como C.

Características

- Modalidad de desarrollo: Grupal.
- Modalidad de entrega: Obligatoria.
- Duración estimada para su desarrollo: 7 semanas.
- Fecha de comienzo: sábado 17/05.
- Fecha de finalización: sábado 05/07.
- Fecha de entrega opcional: sábado 05/07.
- Lugar de corrección: Designado por el ayudante. (Laboratorio o corrección vía mail).

Historial de cambios

Cambios en la versión 1.9

- En la especificación funcional del STT, las palabras "Lista Suspendidos/Bloqueados" se corrigieron por "Lista de Ex-Suspendidos".
- El grupo podrá (si lo desea) agregar las siguientes limitaciones, las cuales deberán poder ser modificadas fácilmente en tiempo de compilación[1]:
    - Máxima cantidad de variables que puede tener un programa.
    - Máxima cantidad de caracteres que puede tener el nombre de una variable/función.
    - Máximo tamaño de un programa (medido en cantidad de bytes que puede tener un archivo .ansisop).
    - Máxima profundidad de llamado entre funciones (medido en cantidad de niveles del stack de funciones).

Cambios en la versión 1.8

- El método de cambio en tiempo de ejecución de los valores de parámetros del sistema queda a criterio del grupo.
- El tiempo de servicio usado en los algoritmos SPN y HRRN será calculado mediante la suma de las *duraciones en segundos* de todas las instrucciones por ejecutar hasta la próxima entrada salida ó finalización del proceso.
- Toda comunicación entre hilos será directa, sin que ningún proceso haga de intermediario.
- La sincronización entre hilos se hará *solamente* mediante semáforos.
- Todas las opciones del script CIDS[2] serán relativas al entorno en el que se está ejecutando.
- Las opciones del script CIDS referentes a configuración se efectuarán antes de iniciar el sistema.
- La opción `detener` del script CIDS se ejecutará desde una consola distinta a aquella en la que se inició el sistema.
- En el lenguaje ANSI Sop, una función puede llamar a otra función.
- En el lenguaje ANSI Sop, el tiempo indicado en la instrucción `io();número` indica la duración de la operación de entrada/salida. La duración de la ejecución de la instrucción en sí misma dependerá del parámetro general.

[1] "Modificadas en tiempo de compilación" significa que si se desea cambiar dichos valores, los mismos deben estar fácilmente accesibles para poder ser modificados y así ejecutar la recompilación del programa.
[2] Configuración, Iniciación y Detención del Sistema.

Desarrollo

Primera parte – Introducción a la programación en C en Linux

A continuación se presentan una serie de ejercicios que intentarán introducir al alumno en el mundo de la programación en C sobre Linux. Los conceptos aquí adquiridos serán de utilidad para el desarrollo de la segunda parte del TP3, y deberán ser complementados con las lecturas de las páginas del manual indicadas y también de tutoriales (encontrados en Internet o los provistos en el grupo google de la materia).

- Compilación de programas en C
  - Codifique un programa básico que lea un número de la entrada estándar y llame a una función llamada f1 que imprima el doble de ese valor. Compílelo y ejecútelo.
  - Investigue como hacer para compilar dicho programa bajo el estándar ANSI C99
  - Declare una variable sin darle uso y vuelva a compilar el programa. Ahora compílelo nuevamente pero con el parámetro –Wall. Investigue los parámetros –Wall y –Wextra de gcc.
  - Investigue brevemente el comando gdb. Depure el programa anterior paso a paso, indagando el estado de sus variables. Investigue si pueden depurarse programas que creen nuevos procesos/hilos.
  - Investigue el comando make. Mueva la función f1 del programa anterior a un nuevo archivo y ahora compile el programa usando make. Investigue como hacer para automatizar algunas tareas comunes como compilar y ejecutar, recompilar, borrar el programa, etc.

- Herramientas de análisis
  - Investigue los comandos ps, pstree y top.
  - Investigue el comando strace. Utilícelo para tracear la ejecución del programa anterior. ¿Por qué no se ve la llamada a la función que imprime el doble del valor?

- Funciones útiles
  - Creación de Procesos: Investigue la función fork(). Modifique el programa anterior para que cree un nuevo proceso que sea el encargado de llamar a la función f1. Haga que dicho proceso también imprima su PID y su PPID.
  - Ejecución de Programas: Investigue la familia exec() de funciones. Modifique el programa anterior para que el nuevo proceso ejecute mediante alguna variante de exec() el binario \bin\ls. Analice las diferencias entre fork() las funciones exec(), así como también los beneficios de combinar ambas funciones.
  - Hilos: Investigue como crear hilos NPTL POSIX. Desarrolle un programa que lea un número de la entrada estándar y lo deposite en una variable compartida. Luego el programa deberá crear dos hilos que impriman infinitamente su identificador de hilo y el valor de dicha variable compartida (entre intervalos de 1 segundo).
  - Ipc (Inter Process Comunication):
    - Realice un programa que cree dos hilos, los cuales se deberán alternar para la lectura y modificación de una variable compartida. Sincronice dichos hilos mediante semáforos POSIX.
    - Investigue como crear segmentos de memoria compartida entre procesos. Realice un programa similar al anterior, pero usando dos procesos en vez de dos hilos.
    - Colas de mensajes (message queues), tuberías (pipes) y FIFOs (named pipes) son otras formas de lograr comunicar procesos o hilos. Téngalos en cuenta para investigarlos en caso de que lo considere necesario.
  - Señales: Investigue qué son las señales en Linux. Modifique el programa anterior para que ante el tecleo de ctrl.+c imprima por pantalla "Usted tecleó ctrl.+c". Ejecute el programa y presione ctrl.+c al menos 2 veces para asegurase que logró el efecto deseado.
  - Errores: Investigue las funciones perror() y strerror(). Haga un programa que intente abrir el archivo "/root/.bashrc" en modo escritura con la función fopen() y que ante un error de la misma llame a perror(). Córralo con el usuario guest.

- Páginas de comandos del **man** recomendadas: gcc, gdb, make, ps, pstree, top, strace, ipcs, ipcrm, mkfifo, kill(1)[3], killall, pkill.

- Páginas de funciones / tutoriales del **man** recomendadas: fork, getpid, getppid, exec, pthreads, pthread_create, sem_overview, mq_overview, ipc(5), pipe(7), fifo(7), select, signal(2), signal(7), kill(2), sleep(3), environ(7).

[3] La notación *nombrepágina(número)* referencia a la página de nombre *nombrepágina* en la sección *número*.

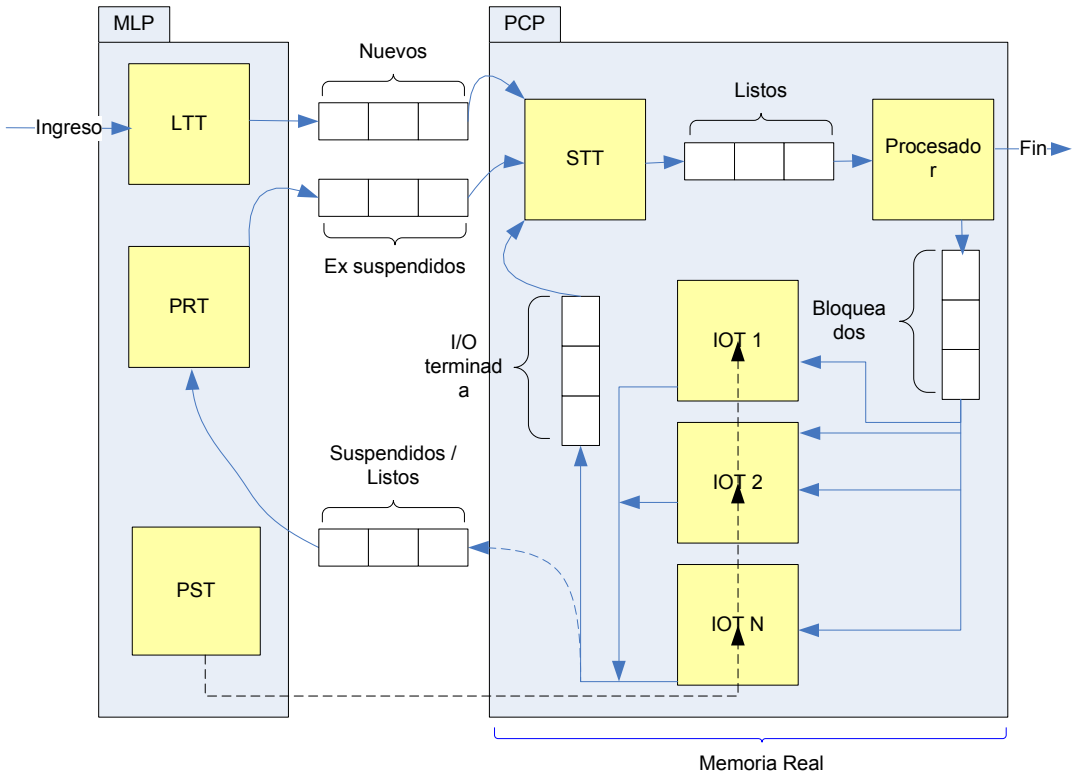
Segunda parte – Resolución de problemas de diseño de Sistemas Operativos

Introducción

El trabajo práctico consiste en el desarrollo de una aplicación que simulará aspectos de un planificador básico de procesos y también el manejo de memoria virtual. Dicha aplicación constará de una serie de módulos que de manera concurrente cumplirán distintas funciones y se sincronizarán entre sí para en su conjunto atacar distintos requisitos que el sistema deberá cumplir.
El sistema tendrá como principal característica la posibilidad de recibir y ejecutar una cantidad variable de programas, los cuales se convertirán en procesos[4] y deberán pasar por los distintos módulos de acuerdo al estado en que se encuentren. Eventualmente, el sistema deberá suspender algunos de ellos, almacenándolos en memoria secundaria para luego reanudarlos cuando sea pertinente.

Arquitectura del Sistema

A continuación se presenta un diagrama que refleja los distintos módulos existentes en el sistema y la interacción entre ellos:



El sistema estará formado por dos procesos[5]. Cada uno de ellos contendrá hilos de ejecución que se sincronizarán y comunicarán con otros hilos, tanto del mismo proceso como del otro. Cuando se encargue la ejecución de un programa, se creará un proceso que será dispuesto en la Lista de Procesos Nuevos.

[4] Dichos procesos no serán procesos reales en Linux, y por lo tanto no serán visibles con comandos como ps/pstree/top.
[5] Dichos procesos serán procesos reales en Linux, visibles con comandos como ps/pstree/top.

Especificación Funcional del Sistema

A continuación se describen los procesos e hilos que constituirán el sistema:

Proceso de Mediano y Largo Plazo (MLP)

Al comenzar su ejecución, deberá iniciar tres hilos: LTT, PRT y TST (explicados en detalle más adelante en este documento), y luego quedará a la espera de directivas del usuario para la ejecución programas. Los mismos serán recibidos desde la entrada estándar y serán derivados al Long Term Thread (LTT) que se encargará de mandarlos a ejecutar cuando corresponda.

Cada programa estará formado por un conjunto de variables, instrucciones y funciones descriptas en un archivo de texto con extensión *.ansisop*[6]. Cuando el usuario decida ejecutar un programa, tipeará "ejecutar *programa*" donde *programa* será la ruta del archivo .ansisop. Por ejemplo: ejecutar /home/guest/programa.ansisop.

En caso de recibir la señal SIGUSR1, deberá avisarle al PST que deberá suspender un proceso cualquiera.

Long Term Thread (LTT)

Este hilo se encargará de recibir del proceso MLP los nuevos programas que el usuario haya encargado. Si la cantidad de procesos iniciados en el sistema no supera un límite preestablecido (llamado MPS) y a su vez el nivel de multiprogramación no llegó a su máximo (llamado MMP), entonces el programa será depositado en la lista de Procesos Nuevos y el hilo STT será notificado para que lo retire cuando sea pertinente. En caso de que no se produzcan las condiciones para que ingresen nuevos programas a ejecutar, los mismos serán almacenados (por orden de llegada) hasta que puedan ingresar a la lista correspondiente.

Process Re-Activation Thread (PRT)

El objetivo de este hilo será tomar el primer proceso de la Lista de Suspendidos/Listos e ingresarlo a la lista de Ex Suspendidos siempre y cuando el nivel de multiprogramación lo permita (se ignorará el MPS). Una vez hecho esto, notificará al hilo STT para que lo retire cuando sea pertinente y tomará nuevamente otro proceso para repetir el ciclo.

Process Suspension Thread (PST)

Cada cierto tiempo, este hilo deberá chequear el estado del sistema para ver si es conveniente suspender[7] algún proceso, permitiendo así que ingresen nuevos programas.

El chequeo del estado consistirá en analizar si la cantidad de procesos iniciados en el sistema no supera el MPS, el MMP ha sido alcanzado, y la lista de procesos listos se encuentra vacía. En caso de cumplirse todas esas condiciones, entonces el PST deberá elegir algunos procesos que se encuentren ejecutando E/S y suspenderlos. Luego, registrará el decremento del nivel de multiprogramación para que el LTT pueda ingresar un nuevo programa a la Lista de Procesos Nuevos.

También deberá suspender un proceso cualquiera (dentro de los que están ejecutando E/S) si recibe una indicación especial del MLP.

Parámetros configurables del proceso MLP

Parámetro[8]	Modificable en tiempo de ejecución[9]
Máximo de procesos que se pueden iniciar en el sistema (MPS).	No
Máximo nivel de multiprogramación (MMP). Este parámetro se refiere a la cantidad de procesos que pueden estar en memoria real al mismo tiempo.	No
Intervalo de tiempo de chequeo del estado del sistema en segundos.	Si
Cantidad máxima de procesos que se pueden suspender en un ciclo.	Si
Ruta del directorio donde se guardarán los procesos suspendidos.	No
Tiempo en segundos que le insume al LTT llevar un programa de una lista a otra.	Si
Tiempo en segundos que le insume al PRT llevar un proceso de una lista a otra.	Si

[6] Al final de este documento se encuentra un anexo que describe la sintaxis y semántica del lenguaje usado para dichos programas.
[7] La suspensión de procesos está descripta en el anexo que detalla la estructura de la imagen de cada proceso.
[8] En la sección "Configuración, Iniciación y Detención del Sistema" se explica como deberá el programa obtener dichos parámetros
[9] El método de cambio en tiempo de ejecución de dichos valores queda a criterio del grupo.

Proceso de Corto Plazo (PCP)

Al comenzar su ejecución, deberá iniciar tres hilos: STT, Procesador y E/S (explicados en detalle más adelante en este documento). Para el caso del hilo de E/S, deberá iniciar tantas instancias de hilos como se haya configurado previamente.

Luego quedará a la espera de la finalización de programas (provenientes del hilo Procesador), notificando dichos eventos por la salida estándar y por un archivo de logueo (se deberán imprimir claramente todas las variables que tenía el programa y los valores con los que quedaron). Todo proceso que se encuentre en el entorno del PCP estará ocupando lugar en memoria real (se incluyen también los procesos de la Lista de Procesos Nuevos y de la lista de Ex Suspendidos, y se excluyen los procesos que hayan iniciado una E/S y luego hayan sido suspendidos).

Short Term Thread (STT)

Cada vez que exista un proceso en alguna de las tres listas posibles (indicadas en el gráfico), este hilo deberá ingresarlo[10] inmediatamente a la Lista de Listos. En caso de haber procesos en más de una lista, tomará todos los de una lista y luego pasará a la siguiente, todo esto en el siguiente orden: Lista E/S Terminada, Lista Ex-Suspendidos, Lista Procesos Nuevos. Existirán los siguientes posibles ordenamientos[11] para la lista de Listos:

- FIFO: Se ordenarán por orden de llegada a esa lista.
- SPN: El tiempo de servicio será el resultante de la suma de las duraciones de todas las instrucciones por ejecutar hasta la próxima entrada salida ó finalización del proceso.
- HRRN: El tiempo de servicio se calculará igual que el obtenido en SPN.

Procesador

Su objetivo será quitar el primer proceso de la lista de Listos, y ejecutar una a una las instrucciones que se encuentren en su código, comenzando desde la última instrucción ejecutada. La ejecución de cada instrucción consistirá en alterar la imagen del proceso[12] de acuerdo al tipo de instrucción leída, insumir el tiempo de ejecución indicado, y pasar a la siguiente instrucción.

En caso de que la instrucción leía corresponda a una E/S, entonces moverá el proceso a la Lista de Bloqueados. Si la instrucción ejecutada es la última, deberá entonces liberar toda la memoria consumida por la imagen del proceso y notificar al PCP enviándole los valores finales de todas las variables que el proceso tenía. Cualquiera sea el caso (E/S o finalización del proceso), deberá luego quitar otro proceso de la Lista de Listos para comenzar/continuar su ejecución.

Input Output Thread (IOT)

Este hilo representará una instancia de Entrada Salida. Tomará el primer proceso de la Lista de Bloqueados y ejecutará su entrada salida, insumiendo el tiempo que corresponda. Una vez finalizado ese tiempo, deberá indagar sobre su estado. Si se encuentra en memoria real, entonces lo depositará al final de la Lista de E/S Terminada y notificará al hilo STT para que lo retire cuando sea pertinente. Si se encuentra en memoria secundaria, entonces lo depositará al final de la lista de Suspendidos/Listos.

Parámetros configurables del proceso PCP

Parámetro	En tiempo de ejecución
Cantidad de instancias del hilo IOT	No
Ordenamiento de la Lista de Listos	No
Intervalo de tiempo (en seg.) que insume cada instrucción de CPU	Si
Intervalo de tiempo (en seg.) que insume cada instrucción de I/O	Si
Tiempo en segundos que le insume al STT llevar un proceso de una lista a otra.	Si

[10] Esto implicará generar la imagen en memoria del programa indicado. Dicha tarea podrá ser realizada por el STT o el LTT.
[11] Dichos algoritmos se encuentran explicados en el capítulo 9: "Planificación uniprocesador" del libro de Stallings
[12] Al final de este documento se encuentra un anexo que describe la estructura de la imagen de un proceso.

Configuración, Iniciación y Detención del Sistema (CIDS)

La configuración, iniciación y detención del sistema se hará mediante un script en bash. A continuación se presentan las funcionalidades que el script deberá soportar:

. do.sh configurar	Pedirá al usuario que ingrese[13] uno por uno los parámetros de configuración requeridos, seteando los valores ingresados por el usuario en variables de entorno[14].
. do.sh configurar iniciar	Ejecutará la acción descripta anteriormente y luego iniciará el sistema solamente si no se encuentra corriendo actualmente.
. do.sh configurar resetear	Le asignará a todas las variables de entorno su valor por defecto.
. do.sh configurar mostrar	Imprimirá por la salida estándar las variables de entorno correspondientes a los parámetros de configuración del sistema, junto con sus valores actuales en el entorno.
. do.sh iniciar	Iniciará el sistema si no se encuentra corriendo actualmente.
. do.sh detener	Se ejecutará desde una nueva terminal y matará a los procesos MLP y PCP enviándoles la señal SIGKILL

Información de estado del Sistema

Logueo del estado de las listas del sistema

Ante la recepción de la señal SIGUSR2 por parte de alguno de los dos procesos (PCP ó MLP), dichos procesos deberán sincronizarse para imprimir el estado de todas las listas del sistema.
Para ello, el proceso que reciba la señal deberá imprimir en un archivo aparte (y si el archivo existe, deberá agregar la información al final) el estado de las listas que a él le competen. Luego le mandará un mensaje al otro proceso para indicarle que haga lo mismo con sus listas. Para lograr que el estado sea (casi) consistente en su totalidad, cada proceso deberá adquirir el uso exclusivo de todas las listas, para luego recorrerlas e imprimirlas.
Ejemplo de un resultado en el archivo:

```
-----------------------------------------
Estado del sistema a las 2008-04-27 18.56.23
-----------------------------------------
Procesos Nuevos: prog1 <- prog2 <- prog3
Ex suspendidos: prog4
Suspendidos Listos: <ninguno>
Listos: prog6
Procesando: prog7
Bloqueados: prog10 <- prog11
I/O 1: prog8
I/O 2: prog9 (suspendido)
I/O 3: <ninguno>
I/O Terminada: <ninguno>
```

Logueo de cambios de listas

Cada vez que un proceso pase de una lista a otra, deberá loguearse dicho evento indicando qué proceso es, de cual lista proviene y a cual lista va. Dichos eventos deberán ser logueados en archivos distintos por proceso, y también deberán ser logueados en un único archivo maestro de cambios de estado de procesos. Por ejemplo, si se ejecutan tres tareas, deberían existir al finalizar su ejecución los archivos: estados_de_todos.log, programa1.estados_log, programa2.estados_log, programa3.estados_log.

[13] En caso de que el usuario presione <enter> sin ingresar valor para un determinado parámetro, se deberá usar un valor por defecto provisto por el propio script.
[14] Como consecuencia, el sistema deberá tomar los parámetros de configuración del entorno.

Anexo A – Alcances y restricciones técnicas

Los procesos MLP y PCP deberán ser binarios separados y uno de ellos deberá iniciar al otro. Los tipos y medios de comunicación entre ellos serán elegidos por el grupo.

Los hilos creados por los procesos deberán ser hilos NPTL (leer "man pthreads")

Los tipos y medios de comunicación entre un proceso y sus hilos serán elegidos por el grupo.

Las listas de Listos, Bloqueados y E/S Terminada deberán estar dispuestas en segmentos de memoria compartida (entre hilos y/o procesos). La sincronización entre hilos se hará solamente mediante semáforos[15], y la implementación a usar será de semáforos POSIX (leer "man sem_overview"). Los procesos no podrán intermediar las comunicaciones entre hilos.

Queda prohibido el uso de sockets en cualquier parte del trabajo práctico.

Anexo B – Lenguaje de programación para los programas: ANSI Sop

Se presenta un ejemplo de un programa codificado en el lenguaje ANSI Sop:

```
# Comentario
variables a,b,c,d,e
comienzo_programa
a=1
b=2;3
c=a+b
d=c-3
f1()
f2()
e=a+c;2
fin_programa
comienzo_funcion f1
a=3
f3()
b=4
fin_funcion f1
comienzo_funcion f2
a=a+1
fin_funcion f2
comienzo_funcion f3
io();2
c=d
fin_funcion f3
```

Se hacen las siguientes aclaraciones:

- Las variables serán globales y solamente de tipo int.
- Existirán dos tipos de instrucciones
  - Asignación: se podrá asignar a una variable alguna de las siguientes variantes
    - Un número entero u otra variable
    - El resultado de una operación aritmética (la cual podrá ser suma o resta, y tendrá solamente dos operandos que podrán ser números o variables)
  - Llamado a una función: podrá ser una función definida por el usuario o la función io() que denota la llamada a un dispositivo de entrada/salida.
- Toda instrucción terminada en *;número* deberá durar *número* segundos, ignorando el valor configurado a modo general (esto no aplica a las llamadas a funciones definidas por el usuario).
- El tiempo indicado en la instrucción `io();número` indica la duración de la operación de entrada/salida. La duración de la ejecución de la instrucción en sí misma dependerá del parámetro general.
- No se proveerán programas que puedan tener errores de sintaxis o semántica en el lenguaje.
- Una función puede llamar a otra función.
- No existirán saltos condicionales ni ciclos en el lenguaje.

[15] Se recomienda leer detalladamente el capítulo 5: "Concurrencia. Exclusión Mutua y Sincronización" y el capítulo 6: "Concurrencia. Interbloqueo e inanición" del libro de Stallings.

Anexo C – Elementos de la imagen de un proceso

La imagen de un proceso son todos aquellos segmentos de memoria que albergan algún tipo de información referida al mismo. A continuación se presenta la imagen de un proceso de este tp:

Elemento	Descripción	Se debe bajar a disco
Datos del programa	Las variables del programa	Si
Código del programa	Instrucciones	Si
Pila del sistema	Pila de llamadas a funciones	Si
PCB	Bloque de control del proceso	No

Cuando el PST decida suspender[16] un proceso, deberá bajar a un archivo de texto con extensión *.imagen* los elementos que deben ser suspendidos. El archivo deberá tener un formato parecido al siguiente:

```
PC=25 (línea de la última instrucción ejecutada)

-------- Código ---------
CÓDIGO DEL PROGRAMA (tal como se encontraba en el .ansisop)
-------------------------

--- Estado Variables ----
a=2
b=0
c=1
-------------------------

-------Estado Pila ------
14,f3 (en la línea 15 se llamó a la función f3)
8,f1 (en la línea 8 se llamó a la función f1)
-------------------------
```

Cuando un proceso de la lista de Ex Suspendidos sea elegido por el STT para pasar a la lista de Listos, se deberá generar nuevamente la imagen basada en dicho archivo y borrar el archivo *.imagen*. La tarea de cargar la imagen a memoria podrá ser llevada a cabo por el PRT ó el STT (quedará a criterio del grupo).

Anexo D – Formato de los archivos de logueo

Toda línea de todo archivo de Log deberá respetar el siguiente formato:

Fecha [NombreProceso] [NombreHilo] [TipoLog]: Data

Donde,

1. **Fecha:** Fecha del sistema, en el formato AAAA-MM-DD HH-MM-SS
2. **NombreProceso:** Nombre del proceso que está escribiendo en el Log, seguido de su pid. Ejemplo: [MLP/5456]
3. **NombreHilo:** Nombre del hilo que está escribiendo en el Log, seguido de su id. Ejemplo: [STT/5456321343]. Esta columna solo aplica a los logueos de hilos
4. **TipoLog:** 'DEBUG', 'INFO', 'WARN' o 'ERROR' según lo que consideren apropiado.
5. **Data**: Detalle del evento registrado con información de contexto si es posible.

[16] Se recomienda leer detalladamente el capítulo 3 del libro Stallings: "Descripción y Control de Procesos"