UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Trabajo práctico Nro. 2

P.R.O.C.E.R.

Planificador de Rutinas Organizadas en Código Entendible y Rústico

Ingeniería en Sistemas de Información
Cátedra de Sistemas Operativos

- 2C2012 -
Versión 1.0

Índice

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[1] 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.

Objetivos de trabajo práctico

Mediante la realización de este trabajo se espera 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 sistemas operativos 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.

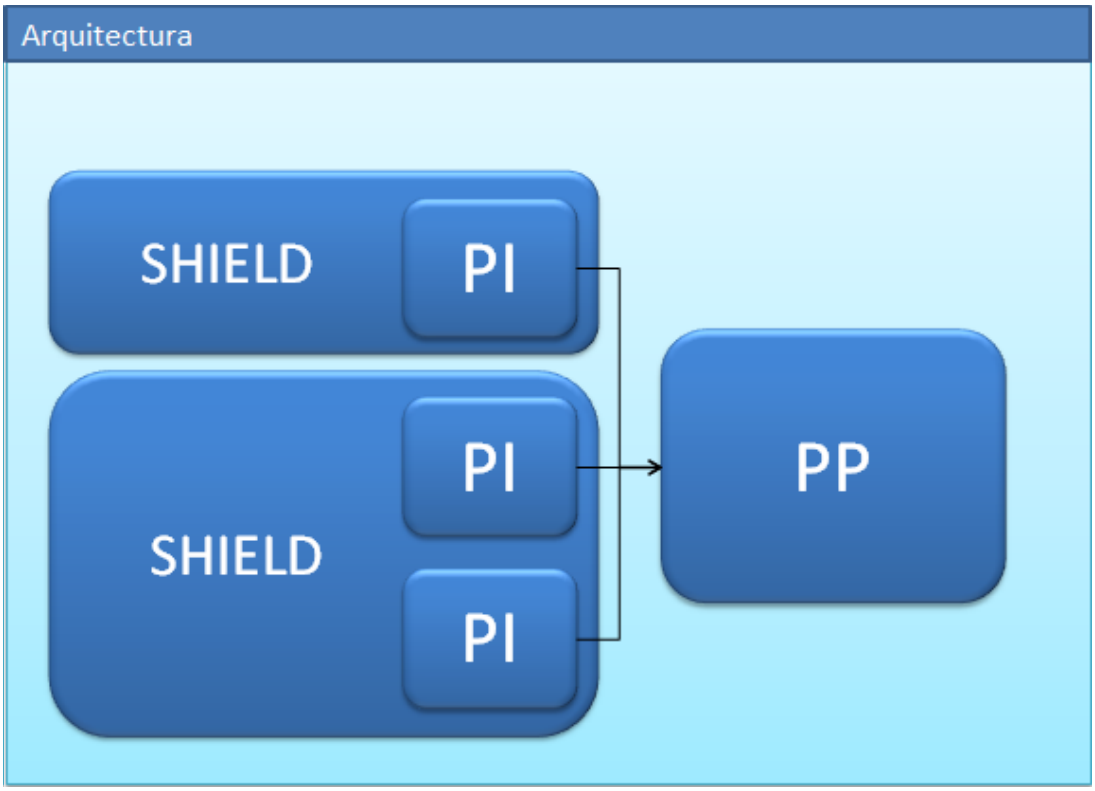
Características

- Modalidad: Grupal y obligatorio.
- Tiempo estimado para su desarrollo: 6 semanas.
- Fecha de comienzo: sábado 13/10.
- Fecha de entrega: sábado 24/11.
- Fecha de primer recuperatorio: sábado 01/12.
- Fecha de segundo recuperatorio: sábado 15/12
- Lugar de corrección: Laboratorio de Medrano

Arquitectura del sistema

1

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



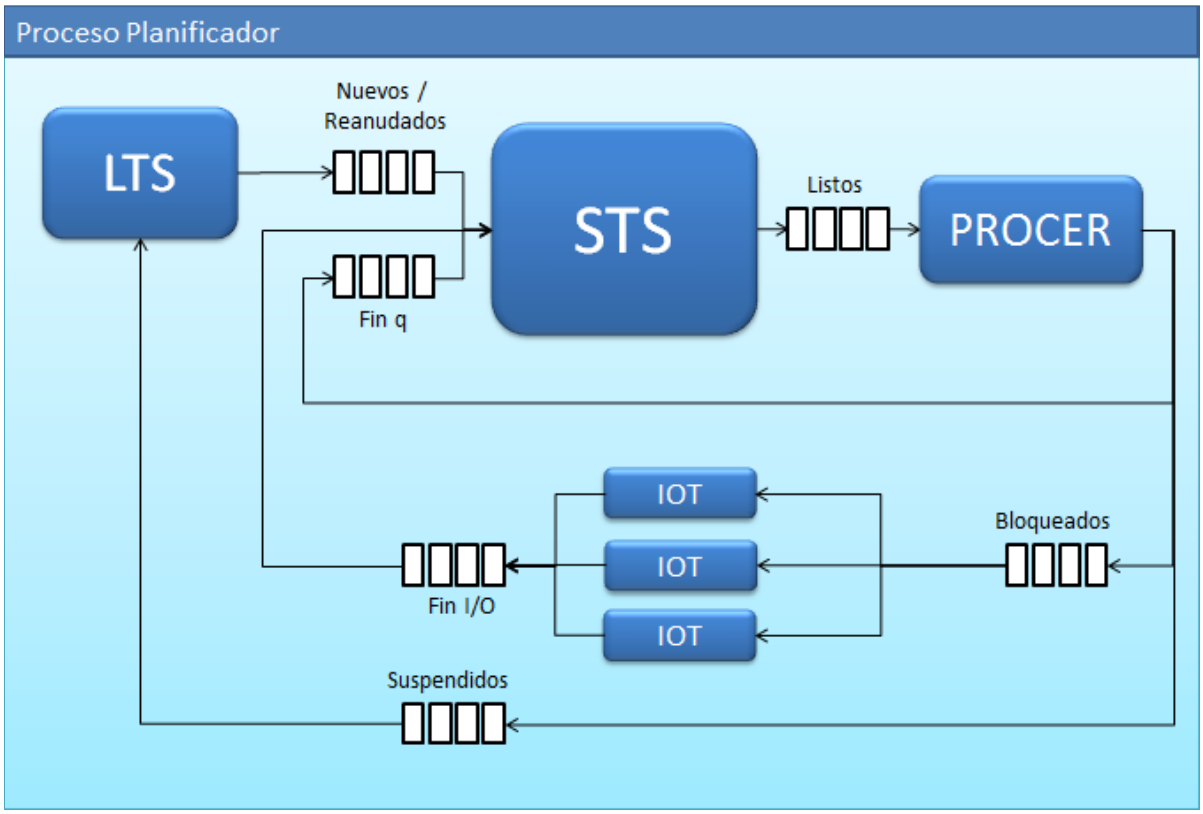
El sistema estará formado por dos procesos. Por un lado el Proceso Planificador (PP) que será el encargado de planificar y ejecutar los procesos, y por otro lado el Proceso Intérprete (PI) que se encargará de iniciarlos, enviándolos al PP, y comunicar al usuario cualquier información significativa.

El Proceso Intérprete como su nombre lo indica será configurado en la cabecera de los scripts AnSISOP y el encargado de permitir su ejecución en el Proceso Planificador

A continuación se detallan los procesos de manera individual y su interacción.

Proceso Planificador (PP)

Al iniciar el proceso este lanzará los hilos LTS, STT, PROCER y la cantidad de hilos IOT configurada. Quedará a la espera de la señal SIGUSR1, y en caso de recibirla deberá suspender el proceso actualmente en el procesador (representado por el hilo PROCER).



Hilo Long Term Scheduler (LTS)

Este hilo al ser lanzado quedará a la espera de nuevas conexiones TCP/IP por parte de Procesos Intérprete (PI). Al recibir una conexión verificará que no haya sido excedido el valor máximo de procesos en sistema (MPS) y que el valor de multiprogramación (MPP) no haya sido alcanzado y le solicitará al proceso intérprete que le envíe el código del nuevo proceso. Una vez recibido el mismo, creará las estructuras de gestión (PCB y segmentos necesarios) y encolará el nuevo proceso en la Lista de Nuevos Procesos (LNP).

En caso de haber alcanzado el grado máximo de multiprogramación, comunicará al proceso intérprete que su ejecución está siendo demorada y encolará dicha solicitud en una lista FIFO hasta que la misma pueda ser atendida.

En caso de superar la cantidad de procesos en el sistema (MPS) notificará al proceso intérprete que el proceso no será iniciado y cerrará la conexión.

Por medio de una señal al proceso, el administrador del sistema puede solicitar que el proceso en ejecución se suspenda y se encole en la lista de procesos suspendidos (LPS). En este caso el LTS le informará a la sesión del PI correspondiente el nuevo estado del proceso, le enviará los valores de sus estructuras de gestión para que este los imprima por pantalla, si correspondiera habilitará otro proceso a que ingrese a la cola de Nuevos respetando las políticas de planificación de largo plazo y quedará a la espera de un mensaje de reanudación para retornar el proceso a la lista de Procesos Reanudados (LPR), siempre y cuando el grado de multiprogramación (MMP) lo permita.

Hilo Short Term Scheduler (STS)

Este hilo será el encargado de verificar las listas de procesos nuevos, reanudados, la lista de finalización de quantum y la de finalización de entrada-salida según el orden de prioridad establecido por archivo de configuración y cargará la lista de procesos listos según el algoritmo definido por archivo de configuración.

Hilo Procesador (PROCER)

Su objetivo será quitar el primer proceso de la lista de Listos, inicializar el contador interno de quantums si corresponde y ejecutar una a una las instrucciones que se encuentren en su código, comenzando desde el inicio o desde la última instrucción ejecutada.

La ejecución de cada instrucción consistirá en alterar la imagen del proceso de acuerdo al tipo de instrucción leída, esperar el tiempo de ejecución indicado (parámetro sleep), y pasar a la siguiente instrucción. Si se tratara de un algoritmo basado en expropiación con quantums de tiempo, cada sentencia incrementará este valor interno en una unidad (es decir, la unidad del quantum será sentencias ejecutadas).

En caso de que la instrucción corresponda a una E/S bloqueante se moverá el proceso a la Lista de Bloqueados. En cambio si la E/S fuera no bloqueante la misma se deberá ejecutar solo si en ese instante existe un IOT disponible. En caso contrario la función deberá devolver un código de error y el programa continuará ejecutando el resto de las sentencias.

Si el proceso finalizara deberá entonces liberar toda la memoria consumida por la imagen del proceso y notificar al LTS enviándole los valores finales de todas las variables que el proceso tenía para que este pueda notificar al PI correspondiente.

Cualquiera sea el caso de salida del proceso del PROCER (E/S o finalización del proceso), el mismo deberá luego quitar otro proceso de la Lista de Listos para comenzar o continuar su ejecución.

Hilo Input Output Thread (IOT)

Este hilo representará una instancia de Entrada/Salida. Tomará el primer proceso de la Lista de Procesos Bloqueados y ejecutará la entrada/salida, insumiendo el tiempo que corresponda. Una vez finalizado ese tiempo moverá el proceso a la lista de Fin I/O y habilitará al hilo STS para que lo retire cuando sea pertinente.

Valores parametrizables del Proceso Planificador

El proceso al iniciar leerá los siguientes parámetros de un archivo de configuración sin un formato especificado.

Parámetro	Modificable en	Valor por defecto

	tiempo de ejecución	
Máximo de procesos que se pueden iniciar en el sistema (MPS).	No	8
Máximo nivel de multiprogramación (MMP)	No	5
Algoritmo de ordenamiento para Lista de Procesos Listos (LPL): *FIFO, RR, PRI*[2]*, SPN*[3]	Si	RR
Quantum de tiempo para algoritmos expropiativos	No	2
Prioridad Lista de Procesos Nuevos (LPN)	Si	3
Prioridad Lista de Procesos Reanudados (LPR)	Si	4
Prioridad Lista de Procesos Fin Q	Si	1
Prioridad Lista de Procesos Fin IO	Si	2
Puerto TCP para recibir conexiones de Procesos Intérpretes	No	5000
Tiempo insumido en segundos por cada sentencia (sleep): *X seg*	Si	1 seg
Tiempo insumido en cada instrucción de I/O: *Y seg*	No	1 seg
Cantidad de instancias del hilo IOT	No	1

Proceso Intérprete (PI)

Este proceso será configurado como intérprete para los archivos AnSISOP[4]. Leerá el contenido del script e intentará conectarse a la IP y Puerto del Proceso Planificador definidas en las variables de entorno PP_IP y PP_Puerto.

Al conectarse enviará el contenido del script y quedará a la espera de información por parte del Proceso Planificador.

Si en algún momento el proceso en ejecución fuera suspendido el proceso intérprete recibirá información sobre el entorno de ejecución. Deberá poder recibir por teclado la solicitud por parte del usuario para reanudar la ejecución del proceso en cuestión.

Cuando el proceso finalice su ejecución, el proceso intérprete recibirá la información de finalización del proceso y la información de su entorno.

Parámetro	Variable	Modificable
IP del Proceso Planificador	PP_IP	No
Puerto del Proceso Planificador	PP_Puerto	No

Anexo - Estructura del proceso

Al igual que un sistema operativo convencional, todo proceso estará identificado por una estructura denominada PCB la cual contendrá el identificador único del proceso, el program

[2] PRI: Algoritmo con prioridades fijas sin desalojo, donde la prioridad se brinda al iniciar el programa

[3] Será SPN sin desalojo, y deberá usar la Fórmula de la media exponencial para estimar las ráfagas del proceso.

[4] Utilizando el hashbang #! y la ruta al binario del proceso intérprete

counter del proceso y las referencias a las tres segmentos fundamentales: datos, código y stack (pila).

Estructura	Información
PCB	Bloque de control del proceso
Segmento de datos	Variables del programa
Segmento de código	Instrucciones del programa
Segmento de stack	Pila de llamadas a funciones

Al suspender un proceso en ejecución el Proceso Intérprete (PI) mostrará por pantalla la información de estado y entorno del proceso, tal y como se muestra a continuación, teniendo en cuenta que los datos entre paréntesis son comentarios del enunciado.

```
----------------------------------------

ID=01 (Identificador de proceso)
PC=25 (Linea de la ultima instrucción ejecutada)

- Estructura de codigo ----
CÓDIGO DEL PROGRAMA (tal como se encontraba en el .ansisop)
-------------------------

- Estructura de Datos ----
a=2 (mostrar nombre de la variable y su valor)
b=0
c=1
-------------------------

- Estructura de Stack ----
14,f3 (en la línea 14 se llamó a la función f3)
8,f1 (en la línea 8 se llamó a la función f1)
----------------------------------------
```

Anexo – Especificación del Lenguaje AnSISOP

El lenguaje de scripting simplificado AnSISOP permitirá que el usuario construya programas y los pueda ejecutar en el sistema. Cabe aclarar que el mismo deberá ser implementado completamente y su especificación no podrá ser ampliada.

Nota: Para la evaluación del trabajo práctico no se proveerán programas con errores de sintaxis.

Sintaxis

- El código principal del programa estará comprendido entre las palabras reservadas *comienzo_programa* y *fin_programa*
- Las sentencias finalizan con un salto de línea. Los saltos adicionales son ignorados.
- Los comentarios se escriben a la derecha de un carácter numeral (#)

- Una palabra seguida de ":" es una etiqueta que será utilizada para permitir saltos dentro del código con *ssc* y *snc*, explicados más adelante.
- Toda sentencia que a continuación tenga un punto y coma y un número deberá ignorar el tiempo por instrucción configurado a modo general, y su tiempo de ejecución será el especificado en la misma línea. Esto no aplica a las llamadas a funciones definidas por el usuario ni a la función *io()*
- Una función puede llamar a otra función.

Variables

Las variables se declaran luego de la sentencia *variables*. Son globales y solamente de tipo entero con signo.

Asignación

Con el nombre de la variable a la izquierda de un signo igual a una variable se le podrá asignar como valor:

- 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.
- El resultado de la función io()

Salto condicional

Las instrucciones de salto condicional *saltar-si-no-es-cero* (snc) y *saltar-si-es-cero* (ssc) recibirán como parámetro una variable que evaluarán y una etiqueta a la que deberán saltar en caso de que se cumpla la condición. Estas instrucciones por definición del enunciado no podrán ser utilizadas dentro de funciones y el origen y el destino del salto deberá ser dentro del código principal del programa.

```
…
i = 1
inicio_for:
i = i + 1
imprimir i
b = i - 10
snc b inicio_for
…
```

Observe que si la variable i no es igual a 10 entonces b = i - 10 no es cero por ende la instrucción de salto condicional retornaría a la sentencia siguiente a la etiqueta *inicio_for*.

Este código de ejemplo incrementará la variable i de uno a diez y la imprimirá dichos valores en pantalla.

Funciones

Las funciones serán declaradas debajo del código principal del programa. Su nombre estará a continuación de la palabra reservada comienzo_funcion y a continuación estará su código hasta la palabra reservada fin_funcion.

io()

Un caso especial es la función io() la cual recibirá dos parámetros entre los paréntesis separados por coma.

1. El tiempo que se accederá al dispositivo
2. El número uno si se trata de una entrada/salida bloqueante y cero si la misma fuera no bloqueante.

Impresión en pantalla

La palabra reservada "imprimir" será utilizada para mostrar el valor de la variable que reciba como parámetro.
El proceso intérprete deberá mostrar en pantalla el nombre de la variable y a continuación su valor.

Ejemplo:

a = 0
imprimir a

Resultado:

IMPRIMIENDO VARIABLE a: 0

Código de ejemplo

```
#!/home/utnso/pi

# 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
```

Anexo – Archivo de Log

Cada proceso tendrá al menos un archivo de log donde registrará la información pertinente sobre el funcionamiento del sistema.

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

```
[Fecha] - [NombreProceso] [NombreHilo] [TipoLog]: [Log]
```

- **Fecha:** Fecha del sistema, en el formato AAAA-MM-DD HH-MM-SS
- **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.
- **TipoLog:** 'DEBUG', 'INFO', 'WARN' o 'ERROR' según lo que consideren apropiado.
- **Log**: Detalle del evento registrado con información de contexto si es posible.

Proceso Intérprete:

Registrará la información relacionada con la ejecución del proceso en un archivo con nombre PI.[pid-del-programa].log

Deberá registrar al menos el inicio y finalización del proceso con su correspondiente información de cierre y en caso de haber sido suspendido la información de estado del mismo.

Proceso Planificador:

Se registrarán los cambios de lista que sufre un proceso, detallando el identificador del mismo, la lista de origen y la de destino utilizando el TipoLog: LSCH

Se registrarán los accesos a entrada/salida no bloqueantes que no hayan podido ser ejecutados, y que proceso estaban utilizando los IOT

Descripción de las entregas

Para permitir una mejor distribución de las tareas y orientar al alumno en el proceso de desarrollo de su trabajo, se definieron una serie de puntos de control y fechas que el alumno podrá utilizar para comparar su grado de avance respecto del esperado.

Checkpoint 1

- Definir las estructuras que el sistema va a utilizar globalmente
- Iniciar el desarrollo del proceso Intérprete. Permitir que el mismo funcione como intérprete en un script ejecutable, que reciba un archivo como parámetro y envíe su contenido por un socket TCP/IP.
- Iniciar el desarrollo del proceso Planificador. Permitir que este reciba conexiones simultáneas de varios PI y las atienda por separado.
- Iniciar el desarrollo del parser de los scripts AnSISOP
- Investigar o desarrollar una biblioteca de colas. Generar un código productor consumidor sobre la misma desde dos hilos sincronizados por semáforos

Tiempo estimado: 2 semanas
Lectura recomendada:

Beej Guide to Network Programming - link
Linux POSIX Threads - link SisopUTNFRBA Common Libraries - link

Checkpoint 2

- Permitir que el Proceso Intérprete reciba de parte del Proceso Planificador la información de los procesos cuando estos sean suspendidos.
- Crear los hilos y las colas del Proceso Planificador y sincronizarlos.
- Iniciar la interpretación de scripts AnSISOP simples.
- Diseñar la lógica de los algorítmos del hilo STS

Tiempo estimado: 2 semanas
Lectura recomendada:

Linux Signals - link Sistemas Operativos, Silberschatz, Galvin - Capítulo 4: HilosSistemas Operativos, Silberschatz, Galvin - Capítulo 5: Planificación del Procesador

Checkpoint 3

- Permitir que un proceso sea iniciado y ejecutado en el sistema
- Habilitar el soporte de funciones en AnSISOP.
- Permitir que un proceso sea suspendido y reanudado.

Tiempo estimado: 1 semanas

Checkpoint 4

- Realizar pruebas de stress en el sistema.
- Realizar los Makefiles correspondientes.
- Validar los requisitos funcionales del trabajo práctico.
- Hacer pruebas de funcionamiento en el laboratorio en la VM server.

Tiempo estimado: 1 semanas