PRACTICA 1: Procesos, *scheduling. IPC*

Ejercicio 1:
Extender el "esqueleto de *shell*" adjunto con la práctica mostrado en clases para que se soporten las siguientes características:

- Colocación de opciones de comando
- Redirección de salida estandar a archivo
- Operador & para llevar un proceso/trabajo a *background*

NOTA: tratar de que la funcionalidad básica sea obtenida, no se pretenden detalles finos.

Ejercicio 2:
Completar el siguiente cuadro sinóptico:

Algoritmo de planificación	Idóneo para sistemas donde predominan procesos	Ventajas	Desventajas o inconvenientes
FCFS			
FSJ			
Prioridades simple			
Round Robin			

Ejercicio 3:
Cinco procesos, ***A***, ***B***, ***C***, ***D*** y ***E***, son lanzados a ejecutar en forma simultánea.
Los tiempos de ejecución en un ambiente de monoprogramación se estiman en 10, 6, 2, 4 y 8 segundos respectivamente. Las prioridades son 3, 1, 4, 5 y 2 respectivamente, siendo 1 la mayor prioridad.

Se supone que los procesos sólo constan de ráfagas de CPU y que la CPU está ociosa en el momento en que se presentan los procesos.

Se desea estimar los tiempos de permanencia en el sistema (tiempo de espera promedio) para cada proceso, ignorando el tiempo de intercambio de CPU entre los mismos (tiempo que lleva el *context switch*), para los siguientes estrategias de planificación:

1. *Round Robin*
2. Planificación por prioridad
3. *First come*, *first served* (FCFS)
4. *Shortest job first* (SJF)

Otras suposiciones:
- En **1** el ambiente es de multiprogramación con una distribución equitativa del CPU (*quantum* fijo de 200 mseg).
- Para **2** y **3** sólo ejecuta de a un proceso por vez, en secuencia.

Ejercicio 4: Shared Memory
Un proceso denominado *director* alloca y escribe en un segmento de shared memory un número entre 0 y 2 el cual hace las veces de identificador de proceso (`ID`), y un número (`NRO`) con valor inicial 1 (en este orden).

Además existen 3 procesos *jugadores*, identificados por un `ID` en [0, 2], que realizan continuamente lo siguiente:

- Leen de la memoria compartida ambos valores de `ID` y `NRO`.
- Si `ID` es el que le corresponde tomará el valor de `NRO` y lo reemplazará por el entero que le sigue que no contenga entre sus cifras a 3 ni sea múltiplo de 3.

- Escribirá por pantalla un mensaje similar a este:
  "Se ejecutó el proceso [ID], se realizó el reemplazo NRO_viejo -> NRO_nuevo"
- Cambiará el valor de `ID` por el del siguiente proceso (secuencia ... 0 1 2 0 1 2 ...)
- Cuando `NRO` alcance el valor de 50, el proceso correspondiente, deberá asignar a `ID` el entero -1. (`ID` del proceso *director*)

Finalmente el proceso *director* deberá liberar la memoria compartida utilizada por los procesos e imprimir por pantalla **"Ciclo finalizado"**

NOTA: en cada intervención de los procesos jugadores estos "attachan" y "dettachan" la memoria compartida al accederla.

Ejercicio 5: Mapped Memory

Un proceso, denominado *porductor*, creaa un archivo con 100 enteros aleatorios en el rango [1,1000]. Posteriormente mapeará el contenido del archivo a memoria.
Otro porceso denominado *sorter*, haciendo uso de `qsort`, ordenará estos enteros (directamente en memoria).
Finalmente *productor* debe mostrar por pantalla el contenido del arachivo ordenado.

NOTA: Las tareas de creación y eliminación del archivo para realizar el mapeo deben quedar bajo la responsabilidad de *productor*.

Ejercicio 6: Pipes

- Crear un proceso que solicite al usuario una operación aritmética sencilla, es decir que involucre solo los operadores binarios **+**, **-**, ***** y /, y que tenga la forma: ***entero operando entero***.
- Este proceso entonces creará un proceso hijo, el mismo recibirá dicha operación mediante un *pipe*, la resolverá y enviará al proceso padre el resultado por *otro pipe*.
- El padre, a continuación, mostrará el resultado de dicha operación por pantalla.
- Finalmente consultará al usuario si pretende realizar otra operación.
  En caso afirmativo se deberán repetir los pasos anteriores.

Utilidades:

- Convertir cadenas a enteros: **`int atoi(const char * cadena)`** de **`<stdlib.h>`**
- Convertir enteros a cadena: **`int sprintf(const char*, char *fmt, …)`** de **`<stdio.h>`**

Ejercicio 7: Uso de FIFOs

Repetir el ejercicio anterior, pero ahora vinculando procesos no relacionados utilizando FIFOs.