**Ejercicios BH**

**Programas concurrentes (padre-hijos-nietos).**

**OBJETIVO.**

Crear varios procesos hijos a partir de un proceso padre que deberá esperar su respectiva terminación.

**DESARROLLO**

En el archivo ***CHeaders.xlsx*** puede usted encontrar algunos de los Headers de C++ para Linux/Unix. Para el caso de ***fork()*** y la familia ***exec…(…)*** lea el archivo ***Fork&Execfam.pdf***.

Dada la situación de ejecución de programas en su directorio de trabajo es fundamental que definan él “.” (punto) al inicio del PATH.

1. Creación de procesos hijos con ***fork()***, y ***exec…(…)***.

Con programas al estilo de clases pasadas, un proceso padre deberá crear tres procesos hijos y esperar a que dichos hijos terminen.

Dentro de este fólder se encuentra el fólder directorio *edoini* donde se encuentran los archivos *padrea.cc*, *hijofina1.cc*, *hijofina2.cc*, y *hijofina3.cc*. Por el momento *padrea* sólo lanza la ejecución del proceso hijo *hijofina1*. Los tres programas hijo tienen programado un ciclo que se repite una cantidad de veces random, esto con el objetivo que al ser ejecutados los procesos hijos terminen en distinto orden, cada vez que se ejecuten todos. También traen programado el mandar ejecutar el comando *ps*.

Entonces el programa *padrea.cc* deberá crear, via sus correspondientes ***fork()***, tres procesos clon hijo, donde cada uno. vía su respectivo ***exec…(…)*** lanzará la substitución por los programas hijos.

Siempre es conveniente que se compilen todos los programas después de modificar alguno de ellos o todos, para esto se les provee con el archivo *pahib.src* que se debe aplicar como *source pahib.src*, logrando así los ejecutables padre e hijos finales.

Antes que nada examine los códigos de los programas.

Siempre se debe imprimir, al final, el valor estatus que envía cada proceso hijo al terminar.

Despliegue un “pantallazo” o “screenshot” de la corrida, ya con los tres hijos. \_\_\_\_\_\_\_\_\_\_\_\_

Dentro del programa ***padrea.cc*** tiene los siguientes argumentos en ***excelp***

execlp("hijofina1.exe","hijofina1.exe","texto-del-programador1",(char \*) 0);

execlp("hijofina2.exe","hijofina2.exe","texto-del-programador2",(char \*) 0);

execlp("hijofina3.exe","hijofina3.exe","texto-del-programador3",(char \*) 0);

donde el string “texto-del-programador#” es pasado al arrancar cada uno de los hijos "hijofina1.exe", "hijofina2.exe" e "hijofina3.exe". Este string “texto-del-programador#” debe ser distinto en cada hijo.

El proceso padre corre en el primer plano.

Los procesos hijos y nietos corren en el background. ¿Qué implicación en el uso de recursos tiene este estado de correr en el background? \_\_\_\_\_\_\_

**SIGUIENTE PÁGINA >>**

Cada hijo final deberá generar un respectivo archivo de texto (hijofina1.txt, hijofina2.txt e hijofina3.txt) donde deberá guardar tanto el string pasado a cada hijo, como el despliegue del comando “ps –l” en cada hijo.

Estos archivos finales de texto los puede generar con las instrucciones de IO File de C++ o puede usar la llamada al sistema *system(…)* como le hicimos en matBG.

De los argumentos del *excelp* antes mostrado, explique para este caso, cual es el significado y la utilidad, de cada uno de los 4 argumentos. \_\_\_\_\_\_\_\_\_\_\_\_

¿En cuál orden terminaron los procesos hijos?\_\_\_\_\_\_\_\_\_\_\_

Muestre el despliegue de los tres archivos de texto.\_\_\_\_\_\_\_\_\_\_

Muestre el árbol de procesos a partir del proceso padre. Diga cuales son los hijos del padre y cuales lo nietos \_\_\_\_\_\_\_\_\_\_\_\_

**<< ANTERIOR PÁGINA**