EVALUACIÓN

EJERCICIO1. GESTIÓN BÁSICA CON SCRIPTS. (0.5 PUNTOS)

En este ejercicio, crearás varios scripts en Shell que te ayudarán a familiarizarte con la gestión de procesos y archivos en un sistema operativo basado en Linux. Cada script debe ser ejecutable y debe seguir las buenas prácticas de programación en Shell.

1.1 **Filtrar Procesos Activos con Mayor Consumo de Memoria**

- **Descripción**: Este script debe listar todos los procesos activos en el sistema y mostrar los que más memoria consumen.
- **Instrucciones**
    1. Creación del script en Bash *consumo_proc.sh*
    2. Abrir el archivo en un editor de texto
    3. Utilizar el comando `ps` para listar los procesos.
    4. Ordenar los procesos por uso de memoria.
    5. Mostrar los N procesos que más memoria consumen, donde N es un número configurable.
    6. Dar permisos de ejecución al script

1.2 Contar archivos y directorios en un directorio

- **Descripción:** De un directorio dado contar el número de directorios y archivos**.** El objetivo de esta práctica es que los alumnos se familiaricen con los siguientes conceptos de sistemas operativos y scripting en Unix:

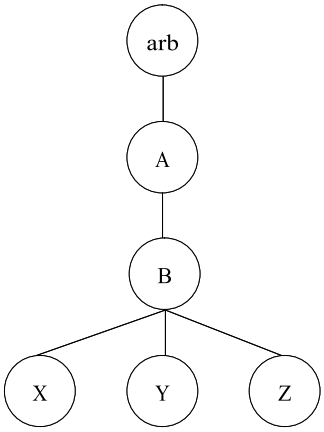
    - Manipulación de archivos y directorios.
    - Verificación de argumentos en un script.
    - Uso de comandos básicos como find y wc.
    - Escritura y ejecución de scripts en Bash**.**

    Al final de la práctica, los estudiantes habrán creado un script que puede contar y mostrar el número de archivos y directorios en una ruta específica.

- **Instrucciones**
    1. Creación del script en Bash *contar_archivos.sh*
    2. Abrir el archivo en un editor de texto
    3. Usa el comando find para encontrar todos los archivos y directorios en el directorio
    4. Usar pipe para pasarlo al comando wc para contarlos
    5. Dar permisos de ejecución al script

EJERCICIO2. GESTIÓN BÁSICA DE PROCESOS. (4 PUNTOS)

Realiza un programa llamado ejec.c que reciba dos argumentos. El programa tendrá que generar el árbol de procesos que se indica y llevar a cabo la funcionalidad que se describe a continuación. El proceso Z, transcurridos los segundos indicados por el segundo argumento, le ordenará mediante una señal al proceso identificado con el primer argumento (A, B, X, Y) que ejecute una orden. Si la señal la recibe el proceso A o B ejecutarán el comando "pstree", Si la señal la recibe el proceso X o Y ejecutarán el comando "ls". El proceso Z no puede utilizar la syscall "sleep" para realizar la espera. Se deberá controlar la correcta destrucción del árbol (los padres no pueden morir antes que los hijos).



El programa debe mostrar por la salida estándar la siguiente información:

```
$ ejec A 15
Soy el proceso ejec: mi pid es 751
Soy el proceso A: mi pid es 752. Mi padre es 751
Soy el proceso B: mi pid es 753. Mi padre es 752. Mi abuelo es 751
Soy el proceso X: mi pid es 754. Mi padre es 753. Mi abuelo es 752. Mi bisabuelo es 751
Soy el proceso Y: mi pid es 755. Mi padre es 753. Mi abuelo es 752. Mi bisabuelo es 751
Soy el proceso Z: mi pid es 756. Mi padre es 753. Mi abuelo es 752. Mi bisabuelo es 751 /*Tras un intervalo de 15 segundos aparecerá*/ Soy el proceso A con pid 752, he recibido la señal.
/*Resultado del comando*/
Soy Z (756) y muero
Soy Y (755) y muero
Soy X (754) y muero
Soy B (753) y muero
Soy A (752) y muero
Soy ejec (751) y muero
```

EJERCICIO2. COMUNICACIÓN ENTRE PROCESOS: TUBERÍAS. (3 PUNTOS)

Realizar un programa llamado copiar.c que permita copiar archivos. El programa recibirá dos argumentos: archivo_origen y archivo_destino, el archivo origen debe existir y el archivo destino se creará. El proceso copiar (proceso padre) generará un proceso hijo y, a partir de ese momento, el proceso padre será el encargado de leer el archivo origen y enviarle la información al proceso hijo que la almacenará en el archivo de destino. La comunicación entre los dos procesos se realizará mediante tuberías (pipe).

```
$ ls
origen.txt
$ copiar origen.txt destino.txt
$ ls
origen.txt destino.txt
$ diff origen.txt destino.txt
$
```

EJERCICIO3. COMUNICACIÓN ENTRE PROCESOS: MEMORIA COMPARTIDA. (2.5 PUNTOS)

Los estudiantes deben crear un programa en C que simule un sistema de procesamiento de datos en paralelo. El programa debe:

- Crear dos procesos: un proceso padre y un proceso hijo utilizando la llamada al sistema fork().
- Utilizar memoria compartida para permitir que el proceso hijo escriba datos y que el proceso padre los lea.
- El proceso hijo generará una lista de diez números aleatorios y los almacenará en un segmento de memoria compartida.
- El proceso padre accederá a la memoria compartida para leer estos números, calcular su media y mostrarla en pantalla.
- Asegurarse de que la memoria compartida se libere adecuadamente después de su uso.

El proceso hijo debe mostrar el siguiente mensaje:

"Soy el hijo (pid) : los números generados son: x, x1,...x9"

El proceso padre debe mostrar:

"Soy el padre (pid). Los números generados fueron: x, x1,...x9. La media es de y"

NORMAS DE ENTREGA

- La entrega de la práctica será durante la semana del 7 al 11 de octubre.
- Se entregará una memoria indicando cómo se han resuelto los distintos ejercicios.
- Se entregarán los códigos fuente de los distintos ejercicios.