Práctica 1

Sistemas operativos

Jordi Blasco Lozano

Sistemas operativos y distribuidos

Grado en Inteligencia Artificial

## Indice:

[Indice: 2](#_Toc179368336)

[1. Ejercicio 1: GESTIÓN BÁSICA CON SCRIPTS 3](#_Toc179368337)

[2. Ejercicio 2: GESTIÓN BÁSICA DE PROCESOS 5](#_Toc179368338)

[3. Ejercicio 3: COMUNICACIÓN ENTRE PROCESOS: TUBERÍAS 6](#_Toc179368339)

[4. Ejercicio 4: COMUNICACIÓN ENTRE PROCESOS: MEMORIA COMPARTIDA 7](#_Toc179368340)

## Ejercicio 1: GESTIÓN BÁSICA CON SCRIPTS

* 1. **Filtrar Procesos Activos con Mayor Consumo de Memoria**

**Comandos usados:** ps, sort, head.

**Implementación:**

Usamos el comando ps -aux para listar todos los procesos.

Luego se ordenan según el uso de memoria con --sort=-%mem.

Se usa head -n N para mostrar los N procesos que más memoria consumen.

A screenshot of a computer code

Description automatically generatedSe añade una línea más  para la descripción de procesos.

A screen shot of a computer screen

Description automatically generated**Resultados:  
(solo tengo 4 procesos activos ya que uso docker)**

* 1. **Contar Archivos y Directorios en un Directorio**

**Comandos usados:** find, wc, tr

**Implementación:**

**if [ $# -eq 0 ]:** comprueba si no hay argumentos para implementar el codigo en el directorio actual si no existe el argumento, y en la ruta que le pasemos si sí que existe el argumento.

ruta="$1": guardamos la ruta en la variable.

find "$ruta" -type d: usamos el comando find para buscar todos los elementos de tipo directorio en la ruta indicada.

| wc -l : pasamos la salida del comando anterior por este comando que devolvera las lineas que haya ejecutado el comando anterior es decir los directorios.

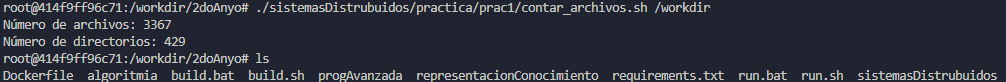
| tr -d '[:space:]': eliminamos cualquier espacio en blanco de la salida para que salga el numero solo.

Lo guardamos en una variable y lo imprimimos, igual que con los archivos.

A screen shot of a computer program

Description automatically generated

**Resultados:**

**Para probar el programa correctamente al usar docker me copio a tiempo real el directorio que tengo en windows en la carpeta workdir de docker, asi que probé a lanzar docker desde una carpeta bastante primitiva a la carpeta donde esta el archivo. Y lo que hay en workdir es todo el primer y segundo año, todo lo que ha contado el programa.**

## Ejercicio 2: GESTIÓN BÁSICA DE PROCESOS

**Comandos usados:** fork, kill, signal, execlp, wait.

**Implementación:**

**Para este ejercicio, las variables se inicializaron como globales para asegurar que cada proceso pudiera acceder a ellas correctamente, ya que cada proceso generado necesita conocer el estado de las demás variables compartidas.**

**Se utilizó un manejador de señales para enviar una señal al proceso especificado como primer parámetro en la ejecución. Esto permite coordinar la ejecución y sincronización entre procesos. Además, se emplearon dos pause() obligatorios en los procesos X y Y para evitar que estos procesos terminen antes de tiempo. La razón de estas pausas es asegurar que los procesos X y Y permanezcan en espera hasta que se les indique que finalicen, lo cual se realiza mediante señales desde el proceso Z después de ejecutar sus funciones (pstree o ps). Esto garantiza que todos los procesos finalicen en el orden correcto.**

**A screenshot of a computer

Description automatically generatedResultados:**

## Ejercicio 3: COMUNICACIÓN ENTRE PROCESOS: TUBERÍAS

**Comandos usados:** fork, pipe, read, write, open, close.

**Implementación**:

**Para este ejercicio se declararon las variables necesarias para abrir la tubería, gestionar los archivos, manejar los bytes leídos, los descriptores y el buffer. Se decidió utilizar variables intermedias para almacenar los resultados de las operaciones de apertura, lectura y escritura, en lugar de usarlas directamente en los condicionales. Esto mejora la legibilidad del código y permite una mejor comprensión de cada operación. Los bucles while se implementaron de manera que iteran continuamente (while true) hasta que se hayan leído y transferido todos los bytes necesarios, utilizando una instrucción break para finalizar el bucle de forma controlada una vez completada la transferencia.**

**En primer lugar, se abre el archivo origen, se crea una tubería y se genera un proceso hijo a partir del proceso padre. Una vez que el proceso hijo está en ejecución, se cierra el extremo de la tubería, ya que el hijo solo necesita leer de esta. Luego se abre el archivo de destino (si no existe, se crea) y se leen todos los bytes de la tubería mientras el proceso padre los está escribiendo simultáneamente en esta. Conforme se leen de la tubería, se van escribiendo en el archivo destino. El proceso padre funciona de forma inversa: lee del archivo origen y escribe en la tubería. Como indicamos al proceso hijo que termine una vez haya completado su tarea, el hijo no repetirá el trabajo del padre, y ambos bucles se ejecutarán al mismo tiempo.**

**Resultados:**

**A screenshot of a computer program

Description automatically generated**A screenshot of a computer program

Description automatically generated

## Ejercicio 4: COMUNICACIÓN ENTRE PROCESOS: MEMORIA COMPARTIDA

**Comandos usados: fork, shmget, shmat, shmdt, shmctl, ftok.**

**Implementacion**:

**Tal como hicimos en el ejercicio anterior, declaramos las variables al inicio y utilizamos más variables de las necesarias para facilitar la comprensión del código. De esta manera, podremos usar estas variables dentro de los condicionales para tener claro qué se está haciendo en cada momento del código.**

**En este ejercicio, vamos a crear un segmento de memoria compartida al cual le asignaremos una variable llamada nums, que será un array de números. Crearemos un proceso hijo que generará una semilla para la generación de números aleatorios y, mediante un bucle que llegará hasta el número de valores deseados (en este caso 10), llenará la variable nums de la memoria compartida con esos números aleatorios.**

**Una vez completado, el proceso hijo terminará, mientras el proceso padre, que estará esperando que su hijo finalice, imprimirá los números de la variable nums del segmento de memoria compartida. Luego de esto, calculará la media de los números.**

**Resultados:**

**A screenshot of a computer

Description automatically generated**