**Sistemas Operativos**

*Prácticas curso 2021-2022*

Programación con llamadas al sistema: Práctica 4

Comandos internos y comandos externos

Hay ciertos comandos de un S.O. que no se pueden ejecutar como comandos externos (programas ejecutables en disco). En su lugar es necesario implementar el comando como un comando interno. Los comandos internos están implementados en el propio intérprete de comandos (bash, csh, etc) y, por tanto, forman parte de su código.

1. El comando ***cd*** (que permite cambiar el directorio por defecto y, en definitiva, navegar por la memoria secundaria) no puede nunca implementarse como comando externo. Para demostrar lo que pasaría si fuera así se ha implementado el comando cd como comando externo. En el directorio **/home/asignaturas/so/Shell** encontrarás el fichero fuente de dicho comando. Cópialo a tu cuenta, observa su código y compílalo. A continuación, ejecútalo con diferentes rutas pero **asegúrate de poner ./cd al ejecutar** (para que no ejecute el comando cd ya implementando en el sistema). ¿Qué observas? Razona y explica por qué ocurre lo que has observado (PISTA: Recuerda que los comandos los ejecuta un hijo…).

No se cambia al directorio especificado porque el que está ejecutando el comando ‘cd’ es un proceso hijo, por lo que el directorio del padre no se ve afectado.

1. En **/home/asignaturas/so/Shell** encontrarás la Version4 del intérprete de comandos, cópialo a tu cuenta. Observa con detenimiento lo que se ha añadido en la función **main()** y la nueva función **ComandoInterno()** que hay antes del **main()**. Contesta a las siguientes preguntas:
   * ¿Qué espera por parámetro y qué retorna la función **ComandoInterno()**?

La función ComandoInterno() devuelve un valor booleano que depende de si el comando introducido al prompt se clasifica como un comando interno o no.

* + ¿Qué se va a implementar en esta función?

En esta función se van a implementar todos los comandos que necesiten ser ejecutados por el proceso padre para cumplir su propósito.

* + Cuando un comando interno (por ejemplo el **cd**) da un error, la función **ComandoInterno()** sigue retornando *true*, razona por qué.

Porque sigue siendo un comando interno, aunque no correcto, por lo que no se creará un proceso hijo que lo ejecute.

* + Enumera los comandos internos implementados en la versión 4.

“cd”, “exit” y “logout”.

* + Supongamos que un comando (por ejemplo el **echo**) se implementa como comando interno en la Version4. Ese comando también es externo (está en **/bin/echo**). ¿Se ejecutaría primero como comando interno o primero como comando externo? ¿Se ejecutaría dos veces?

Solo se ejecutaría una vez y sería ejecutado como comando interno.

* + Los comandos internos se ejecutan con menos recursos y de manera más rápida (son más eficientes). Explica por qué.

Porque no es necesario acceder a un fichero externo. Los comandos internos están implementados directamente en el código del Shell, por lo que es mucho más rápido.

1. Implementa en la Version4 el comando interno **echo** que vuelca a la salida estándar el parámetro indicado a continuación del comando. Sintaxis:

echo [<mensaje>]

Línea 146:

if(strcmp(mi\_argv[0], "echo") == 0) {

// Imprime por pantalla el texto que se le pasa como argumento

for (int i=1; i<mi\_argc; i++) {

cout << mi\_argv[i] << " ";

}

cout << endl;

return true;

}

1. Pon una traza a tu comando interno **echo** y haz pruebas de ejecución con ese comando para ver si has contestado bien al penúltimo apartado del ejercicio 2.

“echo” normalmente no consume el suficiente tiempo como para que lo detecte “time”, por lo que no puede haber mucha diferencia de rendimiento.

1. Implementa en la Version4 el comando interno **prompt** que cambia el **prompt** del intérprete por el indicado a

continuación del comando (PISTA/AYUDA: Haz que la variable ***el\_prompt*** sea global (mueve su declaración fuera del **main()** (al principio del código, bajo los #define)). Sintaxis:

prompt <nuevo\_prompt>

// implementar `prompt`

if(strcmp(mi\_argv[0], "prompt") == 0) {

el\_prompt = mi\_argv[1];

return true;

}