

# Proyecto 1 de Sistemas Operativos.

## Construir una shell simple

Cecilia Hernández

March 24, 2017

**Fecha inicio: Viernes, 24 de Marzo, 2017.**

**Fecha entrega: Martes, 4 de Abril, 2017 (a mediodía).**

### 1. Objetivos

- Fomentar en los estudiantes el desarrollo de habilidades en programación en C o C++.
- Introducir a los estudiantes en el manejo de procesos concurrentes en Unix, creación, ejecución y terminación usando llamadas a sistemas `fork()`, `exec()` y `wait()`. Además el uso de otras llamadas a sistema como `signals`, `pipes`, `getrusage()` para manipular y obtener información de recursos.

### 2. Metodología: Trabajo en grupo de 2 o 3 alumnos.

### 3. Descripción

Desarrollo de un intérprete de comandos simple en Linux (shell). La shell a implementar será similar a las disponibles actualmente en Linux. El desarrollo constará de dos partes de implementación y una de prueba y análisis.

#### **Primera Parte**

La shell debe proporcionar un prompt, lo que identifica el modo de espera de comandos de la shell. Luego, debe leer un comando desde teclado y parsear la entrada para identificar el comando y sus argumentos (debe soportar al menos 3 argumentos). Finalmente debe ejecutar el comando ingresado en un proceso concurrente, para lo cual debe usar el llamado a sistema `fork()` y algunas de las variantes de `exec()`. Los comandos a soportar son ejecutados en foreground, es decir, la shell ejecuta y espera por el término de su ejecución antes de imprimir el prompt para esperar por el siguiente comando. La correcta implementación de esta etapa lo hace merecedor de un 50% de la nota. Nota: Si se presiona "enter" sin previo comando, la shell simplemente imprime nuevamente el prompt.

A continuación se presenta un ejemplo de ejecución.

```

shellABC$> ls -l
total 476
drwxr-xr-x 4 chernandez fcfm 20480 Mar 15 07:33 balancesrc
-rw-r--r-- 1 chernandez fcfm 163840 Jan 7 15:10 balancesrc.tar
drwxr-xr-x 6 chernandez fcfm 4096 Jan 8 09:39 bsponmpi
-rw-r--r-- 1 chernandez fcfm 71680 Jan 7 15:09 denseseq.tar
drwxr-xr-x 2 chernandez fcfm 4096 Jan 7 15:09 src
drwxr-xr-x 4 chernandez fcfm 12288 Mar 4 10:27 vnodesparlocal
drwxr-xr-x 4 chernandez fcfm 12288 Mar 4 18:21 vnodesparsrc
-rw-r--r-- 1 chernandez fcfm 194560 Jan 7 15:11 vnodesparsrc.tar
shellABC$>

```

Su shell debe considerar comandos que contengan pipes, es decir, del tipo *ls -l | grep Jan*, para ello debe utilizar las llamadas a sistema `pipe()`, `dup()` o `dup2()`, and `close()`. Su shell debe soportar múltiples pipes en un comando dado, por ejemplo: *ls -l | grep file | wc -l*

## Segunda Parte

La segunda parte de su shell debe usar el llamado a sistema "getrusage" para permitir recopilar información de los comandos y procesos que ejecuta su shell. Para ello debe crear una estructura de datos que permita almacenar la información recuperada para cada comando ejecutado en la shell, incluyendo comandos repetidos.

Además, su shell debe considerar las siguientes posibles opciones:

- Buscar si un comando fue ejecutado en la sesión y si es así desplegar la información recopilada del uso de recursos.
- Desplegar por pantalla los comandos ejecutados en la sesión como una lista.
- Elegir un comando de la lista y reejecutarlo.

La correcta implementación de esta etapa lo hace merecedor de un 50% de la nota.

## Evaluación

La evaluación final consiste en una nota por el funcionamiento (50%), una por interrogación del grupo (40%) y un informe (10%). El informe debe contener introducción, desarrollo y conclusiones. El desarrollo debe indicar como fueron utilizadas las llamadas a sistema, las estructuras de datos utilizadas en el desarrollo. Esta parte corresponde al 10% de la nota final.