SISTEMAS OPERATIVOS II
Tercer curso Ingeniería Informática.
Curso 2009-2010

Práctica 4: Sistema de ficheros en UNIX: Redirección.

Continuar la codificación de un intérprete de comandos (shell) en UNIX. Al igual que en la práctica anterior

- Los argumentos entre corchetes [] son opcionales.
- Los argumentos separados por | indican que debe ir uno u otro, pero no ambos simultaneamente.
- No debe dilapidar memoria (ejemplo: variable que se asigna cada vez que se llama a una función y no se libera).
- Cuando el shell no pueda ejecutar una acción por algún motivo, debe indicarlo con un mensaje como el que se obtiene con `sys_errlist[errno]` o con *perror()* (por ejemplo, si no puede cambiar de directorio debe indicar por qué).
- En ningún caso debe producir un error de ejecución (segmentation, bus error ...), salvo que se diga explicitamente
- Las direcciones de memoria deben mostrase en **hexadecimal**.
- La información que se nuestra en pantalla no debe incluir en ningun caso líneas en blanco.
- El shell leerá de su entrada estándar y escribirá en su salida estándar, de manera que podría ser ejecutado un archivo de comandos invocando al shell con su entrada estándar redireccionada a dicho archivo.

En esta practica se van a modificar la ejecución en primer plano, la ejecucion en segundo plano y la ejecución sin crear proceso de manera que se permita la redirección de la entrada estándar, la salida estándar y/o el error estándar. Las distintas redirecciones deben ser compatibles entre sí , con la ejecución en segundo plano, la ejecución sin crear proceso, el cambio de prioridad, y la especificación de un entorno. Todas las consideraciones prácticas anteriores sobre la ubicación de ejecutables y la lista de procesos en segundo plano, son de aplicación aquí

[exec] [LISTAVAR] prog [args...] [@pri] [<fich] [>fich] [#fich][&]

- **[exec]][LISTAVAR] prog [args...] [@pri] <fich [&]** Ejecuta (sin crear proceso, creando proceso en primer o segundo plano, con o sin cambio de prioridad, tanto especificándole el entorno como sin hacerlo, según corresponda) el comando *prog* (ejecutable con

parámetros *args*) con la entrada estándar redireccionada al fichero *fich*.

- [**exec**]][**LISTAVAR**] **prog** [**args...**] [**@pri**] **>fich** [&] Ejecuta (sin crear proceso, creando proceso en primer o segundo plano, con o sin cambio de prioridad, tanto especificándole el entorno como sin hacerlo, según corresponda) el comando *prog* (ejecutable con parámetros *args*) con la salida estándar redireccionada al fichero *fich*.
- [**exec**]][**LISTAVAR**] **prog** [**args...**] [**@pri**] **#fich** [&] Ejecuta (sin crear proceso, creando proceso en primer o segundo plano, con o sin cambio de prioridad, tanto especificándole el entorno como sin hacerlo, según corresponda) el comando *prog* (ejecutable con parámetros *args*) con el error estándar redireccionada al fichero *fich*.

  ```
  -> TERM HOME NUEVA=uno a.out arg1 arg2 @15 <f1 >f2 #f3 &
  ```

  ejecuta en segundo plano *a.out arg1 arg2* con un entorno que solo contiene las variables TERM HOME y NUEVA, con la entrada, la salida y el error estándar redireccionados, además establece su prioridad a 15.

pipe com1 % com2 Ejecuta *com1* (ejecutable con parámetros) redireccionando su salida estándar a la entrada estándar de *com2* (ejecutable con parámetros). Tanto *com1* como *com2* representan ejecutables con parámetros. La ejecución es en primer plano

```
#pipe ls -lR . . / % wc -l -c
```

ejecuta ls -lR . . / y redirecciona la salida a la entrada estándar de un proceso que ejecuta wc -l -c

FORMA DE ENTREGA

Como en las prácticas anteriores

FECHA DE ENTREGA VIERNES 11 JUNIO 2010