SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Informática. Curso 2006-2007

Práctica 2: Procesos en UNIX.

Continuar la codificación de un intérprete de comandos (shell) en UNIX, que se irá completando en suscesivas prácticas. Como en la práctica anterior debe tenerse en cuenta que:

- 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 está practica se introducirá la creación de procesos que ejecutan programas (tanto en primer plano como en segundo plano) y la ejecución de programas sin haber creado procesos. **Además, en cualquiera de los casos (ejecución en primer plano, ejecución en segundo plano o ejecución sin crear proceso) para que un ejecutable pueda ser ejecutado debe especificarse la trayectoria completa hasta él (comenzando por "/", "./" o "../") o (en caso de no haberse especificado la trayectoria completa hasta él) debe residir en uno de los directorios de la ruta de búsqueda de nuestro intérprete de comandos (análoga a la variable de entorno PATH en los intérpretes de co-**

mandos mas usuales: bash, ksh . . .) Cada vez que el shell ejecuta un proceso en segundo plano guarda información de dicho proceso en una lista (de procesos en segundo plano lanzados desde nuestro shell). El comando *jobs* permite ver y manipular dicha lista.

getpri [pid] Informa de la prioridad del proceso *pid*. Si no se suministra *pid* informa de la prioridad del shell

setpri [pid] [pri] Establece la prioridad del proceso *pid* a pri. Si no se suministra *pid* establece la prioridad del shell. Si no se especifica ni pid ni pri informa de la prioridad del shell.

uid [id] Si no se especifica uid informa de las credendiales (real y efectiva) de usuario del proceso del shell (informará tanto de la credencial numérica como del login asociado). Si se especifica id establece la credencial efectiva del shell a *id*. *id* puede representar tanto el valor numérico de la credencial como el login asociado. Ejemplo

```
# uid root
  Imposible cambiar credencial: not owner
# uid 15432
  Imposible cambiar credencial: not owner
# uid insaaa00
  Credencial efectiva de usuario establecida insaaa00 (12001)
#uid
  Credencial real de usuario insbbb00 (12007)
  credencial efectiva de usuario insaaa00 (12001)
```

infouser id Informa del usuario *id* (*id* puede ser un número de usuario o un login). Informa del login, uid, gid, nombre, directorio HOME y shell.

search [-a|-d|-n|-p|] [dir|exe]
Manipula la ruta de búsqueda del intérprete de comandos.
La ruta de búsqueda del intérprete de comandos es el conjunto de directorios donde el shell busca los **ejecutables** (Su misión es análoga a la variable de entorno PATH en el shell del sistema). NO DEBE implementarse con una variable de entorno.

search -a [dir] Añade dir a la ruta de búsqueda.

search -d [dir] Elimina dir de la ruta de búsqueda. En estos dos casos, si no se especifica *dir* nos muestra la ruta de búsqueda.

search -n Vacía la ruta de búsqueda.

search -p Añade los directorios de la variable de entorno PATH a la ruta

de búsqueda.

search [dir] Indica si dir está en la ruta de búsqueda.

search Muestra la ruta de búsqueda.

search -f [exe] *exe* representa un ejecutable. Se buscará dicho ejecutable en la ruta de búsqueda, si está nos infomará de la trayectoria completa hasta él. Si *exe* ya representa una trayectoria completa (comenzando por "/", "./" o "../" search comprobará si existe, y en caso de que exista se imprimirá. (ANALOGO AL COMANDO WHICH DEL SISTEMA)

Los directorios se especificarán sin el caracter '/' al final.

```
#search -a /usr/local/bin /* correctos */
#search -a /
#search -a .
......
#search -a /usr/local/bin/ /*incorrecto*/
#search -a ./              /*incorrecto*/
```

```
#search -a /usr/openwin/bin
# search -f xterm
/usr/openwin/bin/xterm
#search -f xternp
 xternp: no encontrado
#search -f ./aa.out
 ./aa.out: no encontrado
#search -f ./a.out
 ./a.out
```

[exec] comando [@pri] [&] El intérprete de comandos ejecuta el programa especificado en *com*. ***com* representa un ejecutable con sus parámetros**. Si se especifica *exec* la ejecución será sin crear proceso (se reemplaza el código del shell). En caso contrario se creará un proceso que es el que ejecuta dicho programa Si el útimo argumento es & la ejecución será en segundo plano, (obviamente el símblo & no debe pasarse como argumento al programa), en caso contario la ejecución será en primer plano. Para poder ser ejecutado, dicho ejecutable debe residir en uno de los directorios de la ruta de búsqueda del intérprete de comandos (*search*) o bien especificarse la trayectoria completa hasta él (comenzando por "/", "./" o "../"). Debe usarse la llamada al sistema *execv*. Si aparece el término *@pri* el programa se ejecutará con su prioridad cambiada a

pri. (obviamente el símblo @ junto con la prioridad no deben pasarse como argumento al programa)
Ejemplos

```
#exec /usr/bin/ls -l /home  /*ejecuta /usr/bin/ls*/
  ....
#exec ./a.out -l /*ejecuta a.out,
                 /*que esta en el directorio actual */
......
#exec ls -l   /* para ejecutar esto, es necesario */
              /*que el directorio  /usr/bin,  */
              /*donde esta el programa ls, se haya incluido */
              /* a la ruta de busqueda */
#/usr/bin/ls -l /home  /*crea un proceso que ejecuta /usr/bin/ls*/
....
#./a.out -l   /* crea un proceso que ejecuta a.out,*/
......        /* que esta en el directorio actual    */
#ls -l        /* para ejecutar esto, es necesario que el directorio */
              /* /usr/bin, donde esta el programa ls, se haya incluido */
              /* a la ruta de busqueda */
#xterm -e csh &  /*crea un proceso que ejecuta xterm  en segundo plano*/
#xterm -e e 10000 @10 & /* crea un proceso que ejecuta xterm  en segundo plan
                 /*pasandole como argumentos -e e 1000 */
                 /*y con la priordad establecida a 10*/
```

jobs [-d] [-n|-s|-a] Muestra una lista de los procesos que se han lanzado en segundo plano desde este shell (dicha lista es mantenida por el propio shell). Para cada proceso debe indicar, el pid, la línea de comando que se está ejecutando, la hora a la que se inició, el estado (activo, terminación normal, terminación debido a una señal), asi como el valor devuelto (o la señal en caso de parado o terminado debido a una señal) y la prioridad que tiene el proceso en ese instante. El listado debe imprimir SOLO UNA LINEA POR CADA PROCESO. Las opciones -n, -s y -a se utilizan para referirse a los procesos que ya hayan terminado: la opción -n a los procesos que hayan terminado normalmente, la opción -s a los que hayan terminado debido a una señal y la opción -a todos los que hayan terminado. La opción -d combinada con una de las otras, indica eliminar procesos de la lista así

```
#jobs /*lista todos los procesos*/
#jobs -a /*lista todos los procesos que han terminado*/
#jobs -s /*lista todos los procesos que han terminado debido a una seal*/
```

```
#jobs  -d -n /*elimna de la lista los procesos que han termniado normalmente
#jobs -d /*lista todos los procesos */
```

proc *pid* Nos da la informacion del proceso *pid* (la misma información que el comando *jobs*). En caso de que el proceso haya terminado nos dará tambien la información ampliada que devuelve *wait4* en la estrcutura *rusage*

```
struct rusage {
   struct timeval ru_utime; /* user time used */
   struct timeval ru_stime; /* system time used */
   long   ru_maxrss;        /* maximum resident set size */
   long   ru_ixrss;         /* integral shared memory size */
   long   ru_idrss;         /* integral unshared data size */
   long   ru_isrss;         /* integral unshared stack size */
   long   ru_minflt;        /* page reclaims */
   long   ru_majflt;        /* page faults */
   long   ru_nswap;         /* swaps */
   long   ru_inblock;       /* block input operations */
   long   ru_oublock;       /* block output operations */
   long   ru_msgsnd;        /* messages sent */
   long   ru_msgrcv;        /* messages received */
   long   ru_nsignals;      /* signals received */
   long   ru_nvcsw;         /* voluntary context switches */
   long   ru_nivcsw;        /* involuntary context switches */
};
```

Información detallada de las llamadas al sistema y las funciones de la librería debe obternerse con man (exec, fork, strtok, wait4, getuid, getpwent, setuid ...)

FORMA DE ENTREGA Va a ser utilizado el servicio de recogida de prácticas suministrado por el Centro de Cálculo de esta Facultad y parte del proceso de corrección de las prácticas va a ser automtico (compilación, listado de practicas entregadas etc) por lo cual deben entregarse **exactamente** como se indica a continuación:

- Se colocará el código fuente de la práctica en el directorio asignado para ello antes de la fecha tope de entrega de la práctica.
- Se entregará UN SOLO fichero fuente por práctica, de nombre pN.c (N el número de práctica). Por ejemplo, para esta práctica será p1.c

(en minúsculas).

- Los grupos de prácticas son de **2 (DOS)** alumnos. La práctica SOLO DEBE SER ENTREGADA POR UNO DE LOS MIEMBROS DEL GRUPO
- en el código fuente de la práctica debe figurar como comentario el nombre de los autores **exactamente** en el siguiente formato

  ```
  /*
  AUTOR:apellido11 apellido12, nombre1:login_en_el_que_se_entrega
  AUTOR:apellido21 apellido22, nombre2:login_en_el_que_se_entrega
  */
  ```

  donde:

  1. La palabra autor aparece en mayúsculas.
  2. Los apellidos y el nombre de los autores están totalmente en minúsculas.
  3. apellidoij representa el apellidoj del componente i del grupo de prácticas.
  4. No hay espacios antes y despues de los dos puntos.
  5. El login que aparece es el del que entrega la práctica (aparece el mismo login en las dos líneas).
  6. Los símbolos de comentarios están en líneas distintas.
  7. No debe incluirse la letra ñ ni vocales acentuadas en los nombres

FECHA DE ENTREGA VIERNES 18 MAYO 2007