SISTEMAS OPERATIVOS

Grado en Informática. Curso 2012-2013

Práctica 2: Procesos

CONTINUAR la codificación de un intérprete de comandos (shell) en UNIX. Nótese que los comandos aquí descritos deben interpretarse de la siguiente manera

- Los argumentos entre corchetes [] son opcionales.
- Los argumentos separados por | indican que debe ir uno u otro, pero no ambos simultaneamente.
- El intérprete de comandos debe aceptar y entender la sintaxis aquí propuesta, pero no tiene que forzarla, por ejemplo, si hay varios argumentos deben aceptarse en el orden especificado, pero puede resultar mas cómodo de programar asumiendo que pueden ir en cualquier orden.

Además deben tenerse en cuenta las siguientes indicaciones:

- **En ningún caso debe producir un error de ejecución (segmentation, bus error . . .)**. La práctica que produzca un error en tiempo de ejecución no será puntuada.
- No debe dilapidar memoria (ejemplo: variable que se asigna cada vez que se llama a una función y no se libera). **NO SE REFIERE A DECLARAR LOS ARRAYS DE TAMAÑO PEQUEÑO** (puede utilizarse *valgrind* para detectar errores de memoria)
- 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).
- El shell leerá de su entrada estándar y escribirá en su salida estándar, de manera que podría (quizá en una práctica futura) ser ejecutado un archivo de comandos invocando al shell con su entrada estándar redireccionada a dicho archivo.

Además de los compandos implememtados el la practica anterior, el shell incorporar ahora los siguientes comandos. El shell llevará una lista de los procesos en segundo plano desde él lanzados (la implementación de la lista es LIBRE). El comando *procesos* nos muestra dicha lista.

prio [**pid** [**valor**]]. Si se especifican dos argumentos (*pid y valor*) se cambiará la prioridad del proceso *pid* a *valor* (utilizando *setpriority*). SI solo

se especifica un argumento se entenderá que es el *pid* y nos mostrará la prioridad de dicho proceso. Si no se suministran argumentos, nos mostrará la prioridad del shell

fork El shell crea un hijo y se queda en espera a que ese hijo termine. (El hijo continua ejecutando el código del shell)

ejecutar prog arg1 ... [@prio] Ejecuta, sin crear proceso (es decir REEMPLAZANDO el código del shell) el programa *prog* con sus argumentos. *prog* representa un ejecutable externo y puede llevar argumentos. En caso de que el último argumento comienze con el caracter `@` se entenderá que el shell debe ejecutar dicho programa con la prioridad cambiada a *prio* y que, obviamente, el programa no debe recibir dicho argumento.

splano prog arg1 ... [@prio] El shell crea un proceso que ejecuta en **segundo plano** el programa *prog* con sus argumentos. *prog* representa un ejecutable externo. En caso de que el último argumento comienze con el caracter `@` se entenderá que dicho programa debe ejecutarse con la prioridad cambiada a *prio* y que, obviamente, el programa no debe recibir dicho argumento. **Además añadirá el proceso a la lista de procesos en segundo plano del shell**

prog arg1 ... [@prio] El shell crea un proceso que ejecuta en primer plano el programa *prog* con sus argumentos. *prog* representa un ejecutable externo. En caso de que el último argumento comienze con el caracter `@` se entenderá que dicho programa debe ejecutarse con la prioridad cambiada a *prio* y que, obviamente, el programa no debe recibir dicho argumento. Por ejemplo, si se desea ejecura el programa `xterm` se escribira en el shell **"xterm"** Y NO **"prog xterm"**

procesos [**all|term|sig|stop|act**] Muestra la lista de procesos en segundo plano del shell.Para cada proceso debe mostrar (en una sola línea) su pid, su prioridad, la linea de comando que ejecuta, el instante de inicio y su estado (activo, terminado normalmente, parado o terminado por señal) indicado, en su caso, el valor devuelto o la señal causante de su terminación o parada. Si se indica *procesos* sin argumentos se listarán todos.

- **procesos** Muestra todos los procesos
- **procesos all** Muestra todos los procesos
- **procesos term** Muestra los procesos terminados normalmente
- **procesos sig** Muestra los procesos terminados por señal
- **procesos stop** Muestra los prcesos parados
- **procesos act** Muestra los procesos activos

borraprocesos [**all|term|sig|stop|act**] Elimina de la lista de procesos en segundo

plano los procesos que se le especifican. (Es un comando de manipulación de la lista de procesos en segundo plano, no tiene que terminar los procesos). Si no se indican argumentos suficientes (p.e. *borraprocesos* sin argumentos) se listarán todos.

- **borraprocesos all** Vacia la lista
- **borraprocesos term** Elimina de la lista los procesos terminados normalmente
- **borraprocesos sig** Elimina de la lista los procesos terminados por señal
- **borraprocesos stop** Elimina de la lista los prcesos parados
- **borraprocesos act** Elimina de la lista los procesos activos

infoproc pid Muestra la información del proceso de pid *pid*. Además de mostrar la información que figura en la lista, en el caso de que el proceso haya finalizado nos da tambien la información de uso de recursos por parte del proceso (`struct rusage`). Debe usarse por tanto la llamda *wait4*

pplano pid Convierte el proceso en segundo plano de identificador *pid* en un proceso en primer plano (esperando a que termine).Cuando termina nos da información de como ha terminado de manera análoga a **infoproc**

Información detallada de las llamadas al sistema y las funciones de la librería puede (y debe) obternerse con man (execvp, fork, wait4, setpriority, ...).

FORMA DE ENTREGA

Las prácticas se entregarán mediante el repositorio de Subversion bajo el directorio P2 antes de proceder a su defensa. Esta carpeta deberá incluir tanto el código fuente como el fichero Makefile (en caso de usarse) que permite su compilación. Las cabeceras de los ficheros fuente deben incluir un comentario con los nombres de los integrantes del grupo de prácticas y el horario en el que están apuntados.

La práctica será entregada y defendida ante el profesor en el aula de prácticas. Todos los miembros del grupo deberán estar presentes para la entrega, de forma que el profesor pueda revisar su funcionamiento así como realizar comentarios/cuestiones a los integrantes del grupo o pedir cambios en el código que se puedan considerar pertinentes.

Las prácticas entregadas que posteriormente no se defiendan con solvencia podrán implicar un `No apto` para todos los miembros del grupo

FECHA LIMITE DE ENTREGA 25-XI-2012