PROGRAMACIÓN AVANZADA

Shell

1. Introducción

En este proyecto final, deberás implementar un intérprete de línea de comando o "shell". El "shell" debe funcionar de manera básica: cuando se escriba un comando, el "shell" crea un proceso secundario que ejecuta el comando que se ingresó y luego solicita más información del usuario cuando finaliza.

El "shell" que implementarás será similar, pero mucho más simple que los que se ejecutan en Unix/Linux todos los días. ¿Cuál es el "shell" que utilizas? Lo puedes averiguar fácilmente ejecutan el comando: `echo $ SHELL`.

El shell se puede ejecutar de dos formas: interactiva y por lotes. En el noto interactivo, mostrará un "prompt" (cualquier cadena que elijas) y el usuario del "shell" escribirá un comando en la solicitud. En modo por lotes, tu "shell" recibe como parámetro de línea de comando un archivo que contiene los comandos a ejecutar. En el modo por lotes, debe repetir cada línea que lea del archivo de texto antes de ejecutarlo; esto te ayudará cuando depures tu "shell". Tanto en modo interactivo como por lotes, tu "shell" dejará de aceptar nuevos comandos cuando encuentre el comando `quit` o llegué al final de la secuencia de entrada (es decir, el final del archivo por lotes o cuando el usuario escriba `CTRL-D`). El "shell" deberá salir después de que todos los procesos en ejecución hayan terminado.

Cada línea (del archivo por lotes o escrita en el "prompt") puede contener múltiples comandos separada con ";". Cada uno de los comandos separados por ";" debe ejecutarse simultáneamente o al mismo tiempo. El "shell" no debe imprimir el siguiente mensaje o recibir más información hasta que todos estos comandos hayan terminado de ejecutar. Por ejemplo, las siguientes línea son todas válidas y tienen comandos razonables especificados:

```
mini-shell>
mini-shell> ls
mini-shell> /bin/ls
mini-shell> ls -l
mini-shell> ls -l ; cat file
mini-shell> ls -l ; cat file ; grep foo file2
```

Por ejemplo, en el último comando, los comandos `ls -l`, `cat file` y `grep foo file2` debe ejecutarse al mismo tiempo; como resultado, puede ser que los resultados de los comando se mezclen.

Para salir del “shell”, el usurio debe teclear `quit`. Esto debería terminar el “shell”, pero ten encuenta que al salir del “shell” no deben estarse ejecutando los otros comando de la misma línea de entrada.

2. Especifícaciones del programa

Tu programa puede ser invocado de la siguiente manera.

```
shell [batchFile]
```

Si el archivo por lotes está presente, tu “shell” debe leer cada línea del archivo para ser ejecutado. Si el archivo no está presente, tu “shell” deberá ejecutar en modo interactivo imprimiendo el “prompt” y leer el comando de `stdin`.

Por ejemplo, si el “shell” se ejecuta...

```
shell myfile
```

Tu programa leer los comandos del archivo `myfile` hasta que encuentre el comando `quit`.

Debes considerar las siguientes situaciones como error; en cada caso, tu “shell” debe imprimir un mensaje de error y salir:

- Un incorrecto número de parámetros.
- Un archivo por lotes que no existe o no puede ser abierto.

Si un comando no existe o no puede ser ejecuta, tu “shell” deber imprimir un mensaje de error **y seguir con su procesamiento**.

Opcionalmente, y con el fin de hacer el desarrollo de tu “shell” más fácil, si recibes una línea de comando con más de 512 caracteres (deberás leer hasta que llegues a salto de línea o 512 caracteres).

Tu “shell” debe ser capaz de manejar los siguientes escenarios sin considerarlos un error:

- Una línea de comando vacía.
- Espacios en blanco extras dentro de una línea de comando.
- El archivo por lotes termina sin el comando `quit` o el usuario **escribe la cadena** `CTRL-D` en modo interactivo.

En ningún caso, ningún comando o línea de entrada puede causar que tu “shell” termine prematuramente. En caos de que no exista un comando entre “;”, debes de imprimir un mensaje de precaución y seguir ejecutando los comandos restantes.

```
mini-shell> ; cat file ; grep foo file2
mini-shell> cat file ; ; grep foo file2
mini-shell> cat file ; ls -l ;
mini-shell> cat file ; ; ; ; ls -l
mini-shell> ; ; ls -l
mini-shell> ;
```

IMPORTANTE: No debe usar la función `system` para ejecutar los comando. Para ello, puedes usar cualquiera de las funciones de la familia `exec`.