Proyecto 1: Mini-Shell

Competencias a Desarrollar

Con este proyecto desarrollará las siguientes competencias:

Competencias teóricas:

• Comprende el proceso de ejecución de programas en Linux y su correspondiente ciclo de vida de procesos.

• Identifica la relación entre las llamadas al sistema de Linux fork y exec y como se aplican en los contextos de ejecución de programas y procesos.

• Describe el concepto de descriptor de archivos, su funcionamiento y su relación con el concepto de archivo en un sistema operativo.

• Comprende el funcionamiento básico del shell dentro de los sistemas operativos tipo UNIX tales como GNU/Linux.

Competencias prácticas:

• Escribe programas en lenguaje C que hacen uso de la interfaz de llamadas al sistema del kernel Linux dentro de los contextos de entrada/salida y ciclo de vida de procesos.

• Prepara distribuciones de código fuente en C para ser compiladas y ejecutadas en sistemas operativos GNU/Linux usando la herramienta GNU Make como sistema de control de compilación.

• Hace uso de las llamadas al sistema de Linux fork, exec, wait, pipe, dup, open, close, read y write en la escritura de programas.

Competencias actitudinales:

• Valora la utilidad y la importancia de usar herramientas de control de compilación en el desarrollo de proyectos.

• Identifica el rol del archivo Makefile y equivalentes en otros sistemas de control de compilación en lo referente a la documentación de proyectos.

Descripción del Proyecto

Este proyecto consiste en el desarrollo de un programa que pueda funcionar como un intérprete de comandos de UNIX básico. Esta clase de programas son conocidos como Shell en la terminología del sistema operativo UNIX.

Como programa, el shell debe proveer una serie de funciones básicas:

• Leer e interpretar lineas de comandos escritas por el usuario.

• Ejecutar los programas indicados en dichas lineas de comandos.

• Esperar a que el programa en ejecución termine antes de seguir leyendo líneas de comandos.

• Identificar operadores especiales y aplicarlos al momento de ejecutar las líneas de comandos.

Su objetivo al implementar este proyecto será entonces desarrollar un programa que provea estas funcionalidades haciendo uso de la biblioteca estándar de C y la interfaz de llamadas al sistema de Linux, sujeto a las siguientes consideraciones:

• El funcionamiento del programa debe centrarse en un ciclo que lea caracteres de la entrada estándar del sistema hasta que ocurra alguna de las siguientes situaciones:

◦ Se reciba un caracter de nueva línea (‘\n’ en sintaxis de C, código ASCII 0x0A). En este caso, su programa debe dejar de leer e interpretar la línea de comandos leída.

▪ Si la línea de comandos es válida, entonces debe ser ejecutada.

▪ En caso contrario, entonces se debe indicar al usuario que hubo un error y volver a leer de la entrada estándar.

◦ Se recibió un caracter o señal de fin de archivo al leer de la entrada estándar.

▪ En este caso se debe mostrar un mensaje de despedida al usuario y terminar la ejecución del programa.

◦ La línea de comandos recibida es solo la palabra “salir”. En este caso igualmente se debe mostrar el mismo mensaje de despedida y terminar la ejecución del shell. La ejecución de los comandos debe realizar mediante las llamadas al sistema fork y exec de Linux. En primera instancia, al determinar que la linea de comandos es válida, su programa debe ejecutar la llamada al sistema fork para crear un sub-proceso, aplicar las redirecciones de entrada/salida apropiadas (ver explicación más adelante en esta sección) y luego aplicar la acción apropiada en los correspondientes procesos como se indica:

•El proceso padre debe ejecutar la llamada al sistema wait para esperar a que el proceso hijo termine su ejecución.

•El proceso hijo debe ejecutar alguna llamada de la familia de llamadas al sistema exec para ejecutar el programa correspondiente identificado en la linea de comandos.

Si el proceso hijo no pudo encontrar el programa a ejecutar, entonces debe reportar el error al usuario y terminar su ejecución inmediatamente, de forma que el proceso padre pueda continuar su ejecución.

Formato de la Linea de Comandos

Su programa debe ser capaz de interpretar lineas de comandos que sigan la siguiente gramática sencilla:

• *comando: programa argumentos*

• *argumentos:*

| *argumento argumentos*

• *linea:*

| *comando '\n'*

| *comando operador comando \n'*

En otras palabras:

• Un comando es el nombre de un programa seguido de una lista de argumentos.

• Una lista de argumentos es o una lista vacía, o un argumento seguido de un espacio en blanco (o tabulador) y luego una lista de argumentos, un argumento siendo una sola palabra. Nótese que esto implica que las listas de argumentos pueden ser arbitrariamente largas. Su programa debe soportar como mínimo listas de argumentos que contengan 20 elementos.

• Una línea de comandos es o una línea vacía (solamente un caracter de nueva línea), o un solo comando, o un comando seguido de un operador y luego otro comando.

Ejemplos de líneas de comandos válidas son las siguientes (la descripción de cada línea se coloca entre paréntesis):

• ls (un (1) comando que es solo un programa sin argumentos).

• ls -l -a ../ (un (1) comando que es solo un programa con 3 argumentos).

• ls -la | wc -l (dos (2) comandos separados por el operador "|").

• mkdir directorio && cp archivo.txt directorio (dos (2) comandos con el operador &&).

Operadores

Su programa debe soportar los siguientes tres (3) operadores:

• "|": se conoce como operador de tubería. Este operador hace que la entrada estándar del programa indicado en el segundo comando de la línea de comandos se conecte con la salida estándar del programa en el primer comando. Por ejemplo, en la línea de comandos "ls -la | wc -l" la salida del programa "ls" debe pasarse como entrada al programa "wc" del segundo comando. Esto puede hacerse mediante una combinación de la función estándar de C "fdopen" y la llamada al sistema "dup" de Linux.

• "&&": se conoce como operador "and". Este operador indica que el segundo comando en una línea de comandos debe ejecutarse si y solamente si el comando anterior en la linea de comandos retornó el código de control cero (0) al proceso del shell. Por ejemplo, en la linea de comandos "uname -a && echo listo", el comando "echo listo" será ejecutado porque el comando "uname -a" se ejecuta correctamente y retorna cero al proceso del shell. Sin embargo, en la línea de comandos "gcc && echo listo", el comando "echo listo" no se ejecutará, porque el comando "gcc" retorna el código de error uno (1) al shell cuando se ejecuta sin argumentos.

• “||”: se conoce como operador “or”. Este operador indica que el segundo comando en una línea de comandos debe ejecutarse si y solamente si el comando anterior en la línea de comandos retornó un código de control distinto de cero al proceso del shell.

En efecto, este operador funciona como el complemento del operador &&. En el caso de los ejemplos anteriores, el comando “echo listo” se ejecutará en el segundo caso, pero no en el primero, al contrario que con el operador &&.

Entregables

Su proyecto debe contener al menos los siguientes archivos al momento de ser entregado para revisión:

• Un archivo README.md (Formato MARKDOWN, ver la sección de Consideraciones).

• Sus archivos de código fuente en C (archivos .c) y cabeceras relacionadas (archivos .h) necesarios para compilar el proyecto.

• Un archivo Makefile que permita compilar el proyecto. El Makefile debe contener al menos las siguientes reglas:

  ◦ all: debe permitir compilar el proyecto completo y producir el programa ejecutable correspondiente.

  ◦ clean: debe eliminar los archivos intermedios de compilación que se produzcan (por ejemplo, los archivos .o) y el ejecutable del proyecto.

• Un informe de a lo sumo 8 páginas en formato .pdf que contenga lo siguiente:

  ◦ Descripción del programa desarrollado. Debe describir como está estructurado el código fuente e identificar los puntos de su código fuente donde se implementaron las funcionalidades que considere más relevantes.

  ◦ Explicación de las llamadas al sistema fork y exec y como son usadas dentro de su implementación del proyecto.

  ◦ Explicación de como implementó las funcionalidades de tuberías y redirección de entrada/salida.

Consideraciones

1. El proyecto debe ser realizado en grupos de dos (2) estudiantes, para fomentar el trabajo en equipo.

2. No se permite el uso de bibliotecas externas. El proyecto debe ser desarrollado usando únicamente la biblioteca estándar de C y las cabeceras del sistema disponibles en GNU/Linux de ser necesarias.

3. Las copias serán penalizadas con la nota mínima para todos los involucrados.

4. El proyecto debe ser entregado a más tardar el día 14-06-2024 a las 11:59 PM, por correo electrónico.

5. El asunto del correo debe ser [SO]_[Proy1]_[Cedula1]_[Cedula2].

6. Los entregables del proyecto deben ser enviados en un archivo .zip al correo laboratoriosisop@gmail.com. El comprimido debe contener un archivo llamado

README.MD que contenga la siguiente información:

- Nombre, apellido y cédula de identidad de los integrantes.
- Pasos para compilar y ejecutar el proyecto.
- Errores conocidos y/o funcionalidades faltantes, de haberlas.