Curso Laboratorio II 2021
5 de mayo de 2020
Programación C

Fecha de entrega: miércoles 26 de mayo de 2021

Es un trabajo **en grupo**. Los grupos fueron previamente sorteados:
Grupos:

A definir

Entregables:

1. Código fuente, bien comentado, bien indentado (4 espacios) con una explicación de qué hace cada función: qué argumentos recibe y qué valor retorna. Debe haber comentarios en aquellas partes de código que no se entienden a primer golpe de vista. **El no cumplir con este punto puede ocasionar la no corrección del ejercicio.**
2. Los programas fuente deberán copiarse en el servidor de la Universidad porque se ejecutarán allí. Es responsabilidad del grupo tener los archivos copiados y haber probado la compilación. Cada grupo tendrá un directorio para copiar sus archivos y generar el ejecutable a entregar. Se dejará un directorio por grupo en:
    - /var/local/labii2021/entregas/minish
3. Debe proveerse un Makefile para la creación del ejecutable con sus dependencias. **Si el ejercicio no compila con "*make minish*" no se corregirá.**
4. Deberá compilarse con las banderas *"-Wall -Wextra -std=gnu99 -ggdb"*. Procuren que la compilación no genere ningún Warning en la versión final.

Los ejercicios serán presentados en clase: un integrante del grupo, elegido por el profesor, mostrará cómo funciona el programa y luego se mostrará parte del código fuente.

Luego de la entrega, habrá una prueba individual en clase que será modificar algo del minish, o agregar alguna funcionalidad nueva, por lo cual todos los integrantes de cada grupo deberán conocer la totalidad del código usado.

Ejercicio minish

El objetivo del ejercicio es escribir una versión mínima de un intérprete de comandos, un *shell* "mínimo" (minish).
minish funcionará como cualquier shell, o sea, un ciclo en el que presenta un "prompt-string", lee una línea de comandos, la interpreta y ejecuta lo que el usuario haya solicitado.
Existe un ejecutable /usr/local/bin/minish para que vean cómo debe comportarse.

Como todo shell, existen comandos internos, programados como parte del shell y que ejecutan sin crear un proceso hijo, además de que deberá poder ejecutar comandos/programas externos a través de la clásica secuencia fork/exec.

Se piden los siguientes comandos **internos** (también llamados "built-in")

- **exit [N]** (terminar el shell, admite un parámetro que es el status de retorno)
- **pid** (no lleva parámetros, muestra el process id)
- **uid** (sin parámetros, muestra el userid como número y también el nombre de usuario) (Pista: getpwuid)
- **gid** (no lleva parámetros, muestra el grupo principal y los grupos secundarios del usuario) (Pista: getgid, getgroups)
- **getenv variable [variable …]** (los parámetros son las variables de ambiente ("environment") para las cuales se quiere saber el valor)
- **setenv variable valor** (define una variable nueva o cambia valor de una variable de ambiente existente)
- **unsetenv var [var…]** (elimina variables de ambiente)
- **cd [dir]** (cambiar el directorio corriente, admite un parámetro)
    - *cd xxx (cambia al directorio xxx)*
    - *cd – (cambia al directorio anterior)*
    - *cd (cambia al directorio de la variable HOME)*
- **status** (sin parámetros: muestra el status de retorno del último comando ejecutado)
- **help [comando]** admite hasta un parámetro que será el comando, para el cual se escribirá una ayuda más extensa. Sin argumentos escribe un texto de ayuda indicando qué comandos internos existen.
- **dir [texto]** (muestra la lista de archivos del directorio corriente, con el agregado de un parámetro opcional que es un texto, en ese caso muestra los archivos cuyo nombre incluya ese texto en cualquier parte del nombre, por ejemplo *dir .c*). Sin argumentos muestra la lista de archivos del directorio corriente).
- **history [N]** (muestra los N – por defecto 10 – comandos anteriores, que deben almacenarse para ejecuciones posteriores del minish, en el archivo $HOME/.minish_history)
- Cualquier otro comando intentará ejecutarse como comando externo.

En **/var/local/labii2021/minish** se proveerá un archivo minish.h que deberán copiar, que contiene las declaraciones de funciones y estructuras a escribir y usar en sus fuentes.

En todos los casos:

- Todas las llamadas al sistema deben controlar que se ejecutaron sin errores y actuar en consecuencia si ocurren. Usar la subrutina ***error*** que vimos en clase.
- El prompt-string deberá tener el usuario y el directorio corriente en él. Por ejemplo: (minish) juan:/var/local/labii2020/publico/minish.h >
- El minish debe atrapar la interrupción del Control-C de teclado (SIGINT), cancelando el ingreso de la línea. Debe recordar ignorar Control-C mientras espera la finalización de un comando externo.

El código tiene que estar estructurado así (esto es OBLIGATORIO, porque permite probar parcialmente el avance del minish, compilando el main con las funciones hechas por el profesor). Se sugiere que cada función esté escrita en un fuente aparte, por ejemplo, el main en minish.c, la función linea2argv en linea2argv.c, etc.

- main

  Loop infinito

  Escribir prompt string en standard error

  Leer una línea y separarla en palabras

  ver función linea2argv() más adelante

  Invocar la función ejecutar

  Actualizar status de retorno

- estructuras importantes a usar:

```
struct builtin_struct {
    char *cmd;                  // nombre del comando interno
    int (*func) (int, char **); // la función que lo ejecuta
    char *help_txt;             // el texto de ayuda
};

extern struct builtin_struct builtin_arr[];
```

- funciones:
  - int linea2argv (char *linea, int argc, char **argv);
    - recibe una línea (texto con un \n final) y hace apuntar argv[0] a la primera palabra, argv[1] a la segunda, etc.
    - argc+1 es el tamaño con el cual está definido argv en el invocador a esta función.
    - Retorna la cantidad N de palabras encontradas y argv[N] deberá ser NULL. Se sobreentiende que N debe ser inferior a argc. Pueden descartarse las palabras “sobrantes”.
    - Una palabra es un texto entre espacios o tabuladores y podrá usarse la comilla o el apostrofe para definir palabras con espacios en el medio.
    - El objetivo de esta función es armar argv para invocar a las funciones anteriores.
  - int ejecutar (int argc, char **argv);
    - decide si es un comando interno, si lo es, lo ejecuta. De lo contrario, invoca a externo().

        - el valor de retorno es el status de retorno del comando.
    - int externo (int argc, char **argv);
        - intenta ejecutar vía fork/execvp, un comando externo
        - el valor de retorno es el status de retorno del comando externo ejecutado.
    - int builtin_XXX (int argc, char **argv);
        - son las funciones que resuelven respectivamente los comandos internos. Son varias funciones, una por cada comando interno, por ejemplo builtin_exit, builtin_pid, builtin_uid, etc.
        - el valor de retorno es el status de retorno (0=éxito).

Para facilitar el avance en la programación, deben avanzar de esta manera:

1. en **la 1ª semana** la programación del main y linea2argv(), con un prompt simple (‘minish> ‘). Cuando ese simple esqueleto funcione, se agregarán las funciones de comandos internos, comenzando por las más simples (help, exit, getpid, getuid, getenv, setenv).
2. En **la 2ª semana** se agrega la ejecución de comandos externos, luego los demás comandos internos y funcionalidad más compleja (como el prompt con el directorio corriente, capturar el retorno del último comando y manejar el Control-C en comandos externos).
3. En **la 3ª semana** se agregan los comandos internos más difíciles que son dir y history. Durante el desarrollo el comando help debe reflejar la realidad de lo avanzado.

Se harán reuniones semanales de seguimiento para ver cómo van en cada semana. La evaluación del trabajo será semanal.

Pistas:

- Funciones de la biblioteca estándar y llamadas al sistema (consultar con *man*):
    - fork, execvp, waitpid, exit
    - read, write
    - getpid, getppid
    - getuid
    - getgid, getgroups
    - getenv, setenv, unsetenv
    - getpwuid
    - getcwd, chdir
    - stat
    - open, openat, fopen, fdopen
    - fgets, fputs, printf, fprintf
    - opendir, readdir
    - sigaction
    - error
    - exit, on_exit

FIN del documento