Trabajo práctico 2: llamadas al sistema en LINUX

Sistemas Operativos

Grupo 18:

Chort, Julio Alberto julioch_17@outlook.com

Pacheco Pilan, Federico Ignacio fedepacheco2112@gmail.com

Reynoso, Valentín valenreynoso17@gmail.com

Segundo Cuatrimestre

2020

1. Resoluciones

1.1. Ejercicio 1: argumentos por línea de comando

opciones.c

```
void vocalEnPosicion(char*, int);
int esVocal(char);                  // En C no existen booleans

int main (int argc, char* argv[])
{
        int i, ch;

        opterr = 0;   // evitar que se imprima el mensaje de error por defecto
        ch = getopt(argc, argv, "lvp:");
        while(ch != -1)
        {
                switch(ch)
                {
                        case 'l':
                                longitudCadena(argv[0]);
                                break;
                        case 'v':
                                vocalesCadena(argv[0]);
                                break;
                        case 'p':
                                vocalEnPosicion(argv[0], atoi(optarg));
                                break;
                        default:
                                printf("Error. %c: opcion no valida.\n", optopt);
                                break;
                }
                printf("\n");

                ch = getopt(argc, argv, "lvp:");
        }

        printf("Cantidad de argumentos: %i\n", argc);

        for(i = 0; i < argc; i++){
                printf("Argumento[%i]: %s\n", i, argv[i]);
        }

        return 0;
}
```

```
void longitudCadena(char* str)
{
	printf("Longitud: %i\n", strlen(str));
}

void vocalesCadena(char* str)
{
	int i;

	for(i = 0; i < strlen(str); i++)
		if (esVocal(str[i]))
			printf("Vocal [%i] = %c\n", i, str[i]);
}

int esVocal(char ch)
{
	char ch2 = tolower(ch);

	if(ch2 == 'a' || ch2 == 'e' || ch2 == 'i' || ch2 == 'o' || ch2 == 'u')
		return 1;
	else
		return 0;
}

void vocalEnPosicion(char* str, int i)
{
	if (i < strlen(str))
		if(esVocal(str[i]))
			printf("Posicion vocal [%i] = %c\n", i, str[i]);
		else
			printf("No es vocal [%i] = %c\n", i, str[i]);
	else
		printf("Error. Posicion fuera de los limites  de la cadena.\n");
}
```

1.2. Ejercicio 2: gestión de procesos

jack.c

```
#include <stdio.h>
#define rutaDoble "./doble"

int main(int argc, char* argv[], char* envp[])
{
	printf("Soy el gran Jack: %i.\n", getpid());
	execve(rutaDoble, argv, envp);
	printf("Toma siguiente, no pudo correrse la escena.\n");     // en caso de
```

```
error

        return 0;
}
```

doble.c

```
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[], char* envp[])
{
        printf("Escena de riesgo: %s = %i.\n", argv[1], strlen(argv[1]));
        printf("Yo %i lo he logrado.\n", getpid());

        return 0;
}
```

1.3. Ejercicio 3: gestión de procesos

valorArbol.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <ctype.h>

void imprimirMensaje(char* hijo, char* padre, int valor);

int main(int argc, char* argv[])
{
        int valor;
        int* aux;

        valor = atoi(argv[1]);
        printf("\nNODO R - VALOR = %i\n", valor);
        printf("ID proceso principal: %i\n\n", getpid());

        if (!fork())    // pid = 0, en el hijo
        {
                valor += 100;
                imprimirMensaje("1", "R", valor);

                if (!fork())
                {
```

```
                        valor *= 2;
                        imprimirMensaje("1.1", "1", valor);
                }
                else
                {
                        wait(aux);    // esperar a que el hijo termine
                        if (!fork())
                        {
                                valor /= 2;
                                imprimirMensaje("1.2", "1", valor);
                        }
                }
        }
        else
        {
                wait(aux);
                if (!fork())
                {
                        valor -= 100;
                        imprimirMensaje("2", "R", valor);
                }
        }

        return 0;
}

void imprimirMensaje(char* hijo, char* padre, int valor)
{
        printf("NODO %s - VALOR = %i\n", hijo, valor);
        printf("ID NODO %s: %i - ID padre NODO %s(NODO %s): %i\n\n", hijo,
getpid(), hijo, padre, getppid());
}
```

1.4. Ejercicio 4: gestión de archivos, directorios y sistema de archivos

listdir.c

```
#include <stdio.h>
#include <unistd.h>
#include <ctype.h>
#include <dirent.h>
#include <string.h>
#define T 128

int mostrarError = 0;

void listarContenidoDirectorio(char* ruta, int esDirectorioActual, int mostrarTipo,
```

```
int mostrarINodo); // en C no hay tipo de dato boolean
void decidirQueListar(char* argvEnOptind, int mostrarTipo, int mostrarINodo);
void decirTipo(unsigned char ch);
void mostrarTitulo(char* ruta, int esDirectorioActual, int mostrarTipo, int
mostrarINodo);
int esPuntoOEsPuntoPunto(char* nombreDirectorio);

int main (int argc, char* argv[])
{
	int i, ch, opciones[3], cantidadOpciones;

	cantidadOpciones = 0;
	ch = getopt(argc, argv, "lti");
	while(ch != -1)
	{
		opciones[cantidadOpciones] = ch;
		cantidadOpciones++;

		ch = getopt(argc, argv, "lti");
	}

	i = 0;
	while((i < cantidadOpciones) && !mostrarError)
	{
		switch(opciones[i])
		{
			case 'l':
				decidirQueListar(argv[optind], 0, 0);
				break;
			case 't':
				decidirQueListar(argv[optind], 1, 0);
				break;
			case 'i':
				decidirQueListar(argv[optind], 0, 1);
				break;
		}

		i++;
	}

	if (mostrarError)
		printf("Error. Directorio no valido.\n");

	return 0;
}

void listarContenidoDirectorio(char* ruta, int esDirectorioActual, int mostrarTipo,
int mostrarINodo)
{
```

```
	DIR* directorio;
	struct dirent* entradaDirectorio;

	directorio = opendir(ruta);
	if(directorio == NULL)
		mostrarError = 1;
	else
	{
		mostrarTitulo(ruta, esDirectorioActual, mostrarTipo,
mostrarINodo);

		entradaDirectorio = readdir(directorio);
		while (entradaDirectorio != NULL)
		{
			if (!esPuntoOEsPuntoPunto(entradaDirectorio -> d_name))
			{
				printf("%s", entradaDirectorio -> d_name);
				if(mostrarINodo)
					printf(" - %i", entradaDirectorio -> d_ino);
				if(mostrarTipo)
				{
					printf(" - ");
					decirTipo(entradaDirectorio -> d_type);
				}
				printf("\n");
			}

			entradaDirectorio = readdir(directorio);
		}

		printf("\n");
	}

}

void decidirQueListar(char* argvEnOptind, int mostrarTipo, int mostrarINodo)
{
	char directorioActual[T];

	getcwd(directorioActual, T);
	if (argvEnOptind == NULL)
		listarContenidoDirectorio(directorioActual, 1, mostrarTipo,
mostrarINodo);
	else
		listarContenidoDirectorio(argvEnOptind, 0, mostrarTipo,
mostrarINodo);
}

void decirTipo (unsigned char ch)
```

```
{
	char str[40];

	switch(ch)
	{
		case DT_REG:
			strcpy(str, "Regular");
			break;
		case DT_DIR:
			strcpy(str, "Directorio");
			break;
		case DT_FIFO:
			strcpy(str, "FIFO");		// nombre dudoso
			break;
		case DT_SOCK:
			strcpy(str, "Socket de dominio local");	// nombre dudoso
			break;
		case DT_CHR:
			strcpy(str, "Especial de dispositivo de caracteres");
			break;
		case DT_BLK:
			strcpy(str, "Especial de dispositivo de bloques");
			break;
		case DT_LNK:
			strcpy(str, "Enlace simbolico");
			break;
	}

	printf("%s", str);
}

void mostrarTitulo(char* ruta, int esDirectorioActual, int mostrarTipo, int
mostrarINodo)
{
	if (esDirectorioActual)
	{
		if (!mostrarTipo && !mostrarINodo)
			printf("Listado del directorio actual:\n");
		else if (!mostrarTipo && mostrarINodo)
			printf("Numero de inodo de archivos del directorio
actual:\n");
		else
			printf("Tipos de archivos del directorio actual:\n");
	}
	else
	{
		if (!mostrarTipo && !mostrarINodo)
			printf("Listado del directorio %s:\n", ruta);
		else if (!mostrarTipo && mostrarINodo)
```

```
                        printf("Numero de inodo de archivos del directorio %s:\n",
ruta);
                else
                        printf("Tipos de archivos del directorio %s:\n", ruta);
        }
}

int esPuntoOEsPuntoPunto(char* nombreDirectorio)
{
        return (!strcmp(".", nombreDirectorio) || !strcmp("..", nombreDirectorio));
}
```

1.5. Ejercicio 5: gestión de señales

galera.c

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#define golpesVarita 3
#define demoraEntregaObsequio 2

void sacarObsequio(int pid, int ObsequioNumero);

int main (int argc, char* argv[])
{
        int obsequioNumero, pidMago;

        srand(time(NULL));
        for(pidMago = atoi(argv[1]), obsequioNumero = 1; obsequioNumero <= 10;
obsequioNumero++)
        {
                sleep(golpesVarita);
                sacarObsequio(pidMago, obsequioNumero);
                sleep(demoraEntregaObsequio);
        }

        return 0;
}

void sacarObsequio(int pid, int obsequioNumero)
{
        if (rand() % 2)    // se usan dos seniales para diferenciar flores de golosinas
        {
                printf("Es una FLOR: %i\n", obsequioNumero);
```

```
                kill(pid, SIGUSR1);
        }
        else
        {
                printf("Es una GOLOSINA: %i\n", obsequioNumero);
                kill(pid, SIGUSR2);
        }
}
```

mago.c

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#define demoraEntregaObsequio 2

int obsequioNumero = 1;

void entregarFlor();
void entregarGolosina();

int main (int argc, char* argv[])
{
        signal(SIGUSR1, &entregarFlor);
        signal(SIGUSR2, &entregarGolosina);

        while (obsequioNumero <= 10)
                /*pause()*/;   // no conveniente: se bloquea el proceso y no puede
verse su pid usando el comando top

        return 0;
}

void entregarFlor()
{
        printf("Ha salido un obsequio para ustedes.\n");
        sleep(demoraEntregaObsequio);
        printf("Dama, esta flor es para usted.(%i)\n", obsequioNumero);
        obsequioNumero++;
}

void entregarGolosina()
{
        printf("Ha salido un obsequio para ustedes.\n");
        sleep(demoraEntregaObsequio);
```

```
        printf("Esta golosina es para vos, niño.(%i)\n", obsequioNumero);
        obsequioNumero++;
}
```

1.6. Ejercicio 6: redirección de entrada y salida

usuario.c

```
#include <stdio.h>
#define T 128

int main ()
{
        char str[T];

        while(scanf("%s", &str) == 1)    // scanf retorna el numero de variables que
pudo leer correctamente
                printf("u%s\n", str);

        return 0;
}
```

clave.c

```
#include <stdio.h>
#include <string.h>
#define numeroFijo "2718" // itoa no funciona (no olvidar las comillas)
#define T 128

int main ()
{
        char str[T];

        while(scanf("%s", &str) == 1)    // scanf retorna el numero de variables que
pudo leer correctamente
        {
                strcat(str, numeroFijo);
                printf("%s\n", str);
        }

        return 0;
}
```

habilitar.c

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
#define rutaUsuario          "./usuario"
#define rutaClave            "./clave"
#define rutaNombresTxt   "./nombres.txt"
#define rutaUsuariosTxt    "./usuarios.txt"

int main (int argc, char* argv[], char* envp[])
{
        int tuberia[2];
        int* aux;

        pipe(tuberia);

        if (!fork())                                    // en el hijo
        {
                close(0);                                       // cerrar entrada estandar
                open(rutaNombresTxt, O_RDONLY);                 // entrada desde
nombres.txt

                close(tuberia[0]);                              // cerrar lectura tuberia
                close(1);                                       // cerrar salida estandar
                dup(tuberia[1]);                                // salida a la tuberia

                execve(rutaUsuario, argv, envp);            // ./usuario ...
        }
        else                                                    // en el padre
        {
                wait(aux);

                close(0);
                dup(tuberia[0]);                               // entrada desde tuberia

                close(tuberia[1]);                             // cerrar escritura tuberia
                close(1);
                open(rutaUsuariosTxt, O_CREAT | O_WRONLY);        // salida a
usuarios.txt

                execve(rutaClave, argv, envp);                      // ./clave ...
        }

        return 0;
}
```

2. Consideraciones

2.1. De la resolución:

Ejercicio 1: -

Ejercicio 2: -

Ejercicio 3: -

Ejercicio 4:

- Se optó por no mostrar por pantalla las entradas "." y ".." porque no figuraban en los ejemplos de ejecución.
- A cada opción y 'tipo' de directorio (actual o no) se le asignó un 'título' distinto en la salida.
- No estamos seguros de los nombres que colocamos a algunos de los tipos de archivos: *FIFO* (DT_FIFO) y *Socket de dominio local* (DT_SOCK). Esto es así porque en el enunciado no se mencionan explícitamente.
- La sentencia *struct dirent* entradaDirectorio;* dentro de la función *listarContenidoDirectorio()* no estamos seguros porqué solo funciona de esa manera, más concretamente porqué debe colocarse *struct* delante.

Ejercicio 5:

- Parece que en algunas ejecuciones no se demora exactamente dos segundos el mago en entregar un obsequio o, luego de esto, tres segundos en sacar el próximo. No sabemos si este problema es 'real', y en tal caso no se nos ha ocurrido una solución razonablemente sencilla.
- No entendemos muy bien a qué se refiere el enunciado cuando se dice "(...) lo que toma fuertemente la galera a modo de señal para extraer la próxima (...)".

Ejercicio 6: -

2.2. Fuentes consultadas:[1]

Ejercicio 1:

[1] En general, no se citan las entradas de manual consultadas. No se cita el material proporcionado por la cátedra.

- http://www.cplusplus.com/reference/cstring/
- http://www.cplusplus.com/reference/cctype/tolower/
- http://www.cplusplus.com/reference/cstdlib/atoi/

Ejercicio 2: -

Ejercicio 3: -

Ejercicio 4:

- https://stackoverflow.com/questions/4204666/how-to-list-files-in-a-directory-in-a-c-program
- https://stackoverflow.com/questions/23958040/checking-if-a-dir-entry-returned-by-readdir-is-a-directory-link-or-file-dent
- https://www.gnu.org/software/libc/manual/html_node/Directory-Entries.html
- http://www.cplusplus.com/reference/cstring/strcpy/
- http://www.cplusplus.com/reference/cstring/strcmp/
- https://stackoverflow.com/questions/298510/how-to-get-the-current-directory-in-a-c-program

Ejercicio 5:

- http://www.cplusplus.com/reference/cstdlib/srand/
- http://www.cplusplus.com/reference/cstdlib/rand/
- https://www.gnu.org/software/libc/manual/html_node/Miscellaneous-Signals.html

Ejercicio 6:

- http://www.cplusplus.com/reference/cstdio/scanf/
- http://www.cplusplus.com/reference/cstdlib/itoa/
- https://stackoverflow.com/questions/15291523/accepting-any-number-of-inputs-from-scanf-function
- https://stackoverflow.com/questions/16726377/o-wronly-undeclared-first-use-in-this-function

Misceláneo:

- http://www.cplusplus.com/reference/cstdio/printf/