Tarea 1

Nombre del Alumno: Cabrera Garibaldi Hernán Galileo

Nombre del profesor: Carlos Alberto Román Zamitis

Clave de la Materia: 2946

Nombre de la Materia: Arquitectura Cliente Servidor

Semestre: 2021-1

Ejercicio 1 - ids.c

Código

```
#include<stdio.h>
#include<unistd.h>
#inlcude<stdlib.h>
int main(void){
  printf("REal user ID: %d\n", getuid());
  printf("Effective User ID: %d\n",getuid());
  printf("Real group ID: %d\n", getgid());
  printf("Effective group ID: %d\n", getgid());
  return 0;
}
```

Ejecución

```
galigaribaldi@MacBook-Pro-de-Hernan Tarea % nano ids.c
galigaribaldi@MacBook-Pro-de-Hernan Tarea % gcc ids.c -o id
galigaribaldi@MacBook-Pro-de-Hernan Tarea % ./id
REal user ID: 501
Effective User ID: 501
Real group ID: 20
Effective group ID: 20
galigaribaldi@MacBook-Pro-de-Hernan Tarea %
```

Ejercicio 2 - child.c

Código

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

```c
#include <stdlib.h>
#include<unistd.h>
int main(void) { //Tipo de dato que identifica el proceso del Hijo
  pid_t child;
  if((child = fork()) == -1) {
    //fork() Devuelve valores entero
    //Cero: Se ha creado el proceso Hijo
    //Valor positivo: Se devuelve al proceso padre a quien lo mande a llamar
    //Valor engativo: Fallo la creación del proceso Hijo
    perror("fork"); //Mensaje de error
    exit(EXIT_FAILURE); //Finaliza la llamada del rpoceso
  }
  else if(child == 0) { //Cuando chil es igual a o significa que el proceso ha
concluido
    puts("in child");
    printf("\tchild pid = %d\n", getpid());//Id del proceso
    printf("\tchild ppid = %d\n", getppid());//ID del proceso
    exit(EXIT_SUCCESS);//Finaliza la ejecucion exitosa
  } else {
    puts("in parent");
    printf("\tparent pid = %d\n",
    getpid());
    printf("\tparent ppid = %d\n", getppid());
  }
  exit(EXIT_SUCCESS);//Finaliza la ejecucion
}
```

Ejecución

```
galigaribaldi@MacBook-Pro-de-Hernan Tarea % gcc child.c
galigaribaldi@MacBook-Pro-de-Hernan Tarea % gcc child.c -o child
galigaribaldi@MacBook-Pro-de-Hernan Tarea % ./child
in parent
        parent pid = 5043
        parent ppid = 4847
in child
        child pid = 5044
        child ppid = 1
galigaribaldi@MacBook-Pro-de-Hernan Tarea %
```

if((child = fork())==-1)

fork() devuelve un valor negativo en el caso de que la creación del proceso hijo no tenga éxito y devuelve los siguientes valores en los diferentes casos:

- *Cero*: Se ha creado el proceos Hijo
- *Valor positivo:* Se devuelve al procesos padre o quién lo llama(El valor contiene el ID del proceso Hijo recién creado)

- *Valor Negativo:* Falló la creación del proceso Hijo

Perror()

La función *perror* transforma el número de error en la expresión entera de errno a un mensaje de error. Escribe una secuencia de caracteres al stream estándar de errores, esto es: primero (si cadena no es un puntero nulo y el carácter apuntado por cadena no es el carácter nulo), la cadena apuntada por cadena seguido de dos puntos (:) y un espacio; entonces un mensaje de errores apropiado seguido por un carácter de línea nueva. El contenido de los mensajes de errores es el mismo que aquello retornado por la función strerror con el argumento errno, que están definidos según la implementación.

perror("fork")

Función que imprime un mensaje de error descriptivo en cadena (contiene mensaje personalizado que se imprimirá antes del mensaje de error).

Función exit() y sus variantes

La función de biblioteca C void exit (int status) finaliza el proceso de llamada inmediatamente. Todos los descriptores de archivos abiertos que pertenecen al proceso se cierran y los hijos del proceso son heredados por el proceso 1, init, y el padre del proceso recibe una señal SIGCHLD.

- **EXIT_SUCCESS:** Esta macro se puede utilizar con la función de salida para indicar la finalización exitosa del programa. En los sistemas POSIX, el valor de esta macro es 0. En otros sistemas, el valor puede ser alguna otra expresión entera (posiblemente no constante).
- **EXIT_FAILURE:** Esta macro se puede utilizar con la función de salida para indicar la finalización incorrecta del programa en un sentido general. En los sistemas POSIX, el valor de esta macro es 1. En otros sistemas, el valor puede ser alguna otra expresión entera (posiblemente no constante). Otros valores de estado distintos de cero también indican fallas. Ciertos programas utilizan diferentes valores de estado distintos de cero para indicar tipos particulares de "no éxito". Por ejemplo, diff usa el valor de estado 1 para indicar que los archivos son diferentes y 2 o más para indicar que hubo dificultades para abrir los archivos.

BIBLIOGRAFÍA

Anónimo. (2016). Exit Status. 2020, de gnu.org Sitio web: https://www.gnu.org/software/libc/manual/html_node/Exit-Status.html

Anónimo. (2016). C library Function. 2020, de tutorialspoint Sitio web: https://www.tutorialspoint.com/c_standard_library/c_function_exit.htm

Max. (2016). Funcionamiento Perror. 2020, de Bibliotecas ANSI C Sitio web: http://c.conclase.net/librerias/?ansifun=perror