Instituto Politécnico Nacional

Escuela Superior de Cómputo

Sistemas Operativos

Bash con fork, exec y wait

Integrantes:

- Serrano Carreño Juan Ángel
- Lozano Amaro Jaime Armando

Grupo 4CM1

Profesora Pérez De Los Santos Mondragón Tanibet

Código

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <errno.h>

#define NUM_COMMANDS 6

const char *valid_commands[NUM_COMMANDS] = {
    "printPpid",
    "printPid",
    "printGroup",
    "printSesion",
    "printUser",
    "printSerieFibonacci"
};

int isValidCommand(char *command) {
    char *cmd = strtok(strdup(command), " "); // Duplica y extrae el primer token

    // Compruebq lo duplicado con los comandos válidos
    for (int i = 0; i < NUM_COMMANDS; i++) {
        if (strcmp(cmd, valid_commands[i]) == 0) {
            return 1;
        }
    }
    return 0;
}

void executeCommand(char *command) {
    char *args[3];
    char execPath[1024];

    args[0] = strtok(strdup(command), " "); // Duplica y extrae el primer token
    args[1] = strtok(NULL, " "); // Extrae el segundo token si existe
    args[2] = NULL; // Finaliza el array de argumentos

    // Construye la ruta del ejecutable (mismo directorio)
    snprintf(execPath, sizeof(execPath), "./%s", args[0]);

    pid_t pid = fork();
```

```
    if (pid == 0) { // Proceso hijo
        //Usando execvp para ejecutar el comando obtenido del usuario
        //execvp busca el comando en los directorios del PATH del sistema pero si
se le pasa la ruta del ejecutable, se ejecutará el comando
        if (execvp(execPath, args) == -1) {
            fprintf(stderr, "Error al ejecutar: %s\n", strerror(errno));
        }
        exit(EXIT_FAILURE); // Salida del hijo si exec falla
    } else if (pid < 0) {
        //strerror convierte el código de error en una cadena de error,. En este
caso, el código de error es errno y lo manda a stderr
        fprintf(stderr, "Fallo en fork: %s\n", strerror(errno));
    } else { // Proceso padre
        int status;
        waitpid(pid, &status, 0); // Espera a que el hijo termine
        //Se usa WIFEXITED para comprobar si el hijo terminó normalmente y
WEXITSTATUS para obtener el estado de salida del hijo
        if (WIFEXITED(status) && WEXITSTATUS(status) != 0) {
            printf("El comando falló con el estado %d\n", WEXITSTATUS(status));
        }
    }
}

int main() {
    char command[100];

    while (1) {
        printf("$ ");

        // Obtiene lo que el usuario escriba en la consola
        if (fgets(command, sizeof(command), stdin) == NULL) {
            break;
        }

        // Remueve el salto de línea
        command[strcspn(command, "\n")] = 0;

        // Salir del shell
        if (strcmp(command, "EXIT") == 0) {
            break;
        }

        // Comprueba si el comando es válido
        if (isValidCommand(command)) {
            executeCommand(command);
```

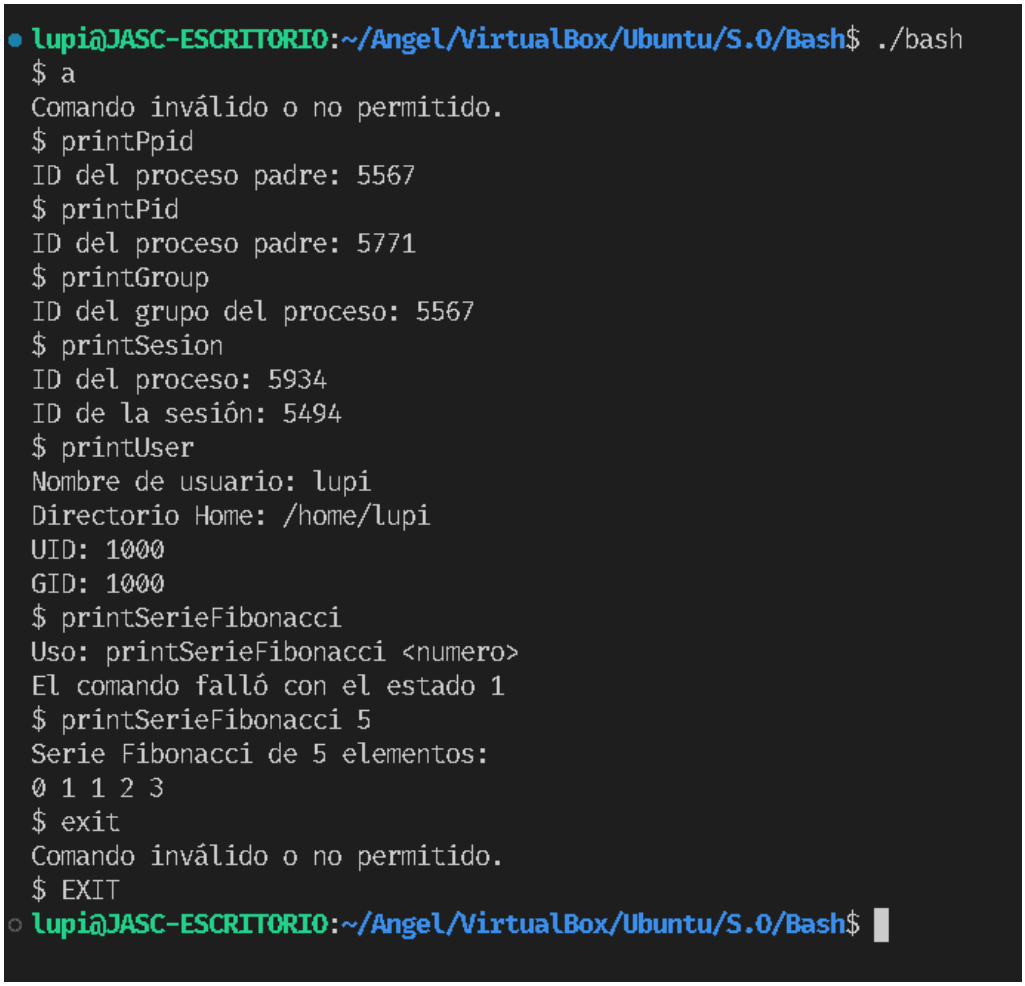
```
        } else {
            printf("Comando inválido o no permitido.\n");
        }
    }

    return 0;
}
```

Concepto general

Nuestro programa en un ciclo while infinito hace el siguiente proceso:

1-Imprime el signo de $
2-Atravez de un fgets obtiene lo escrito por la stdin y lo guarda en una variable
3-Remueve el carácter de termino de línea de nuestra variable
4-Si esta cadena guardada es EXIT termina el programa
5-Comprueba si la cadena guardada es un comando, para esto lo divide a través de los espacios y lo pone en un arreglo, el primer argumento es el comando
6-Si el comando esta en la lista de comando aceptado entonces si existe en caso contrario regresa un error
7-Si el comando es valido con el comando se construye la ubicación relativa del programa (este se debe encontrar en misma carpeta que el bash)
8-Se crea un fork, el padre usa wait para esperar a que termine el hijo y con WIFEXITED detecta si el hijo no tuvo ningún error
9-El hijo con execvp y con el path creado lo manda a ejecutar el programa, esto ejecutara los ejecutables ya hechos, si hay algún error en este proceso se imprime y se termina el proceso del hijo.

Funcionamiento


lupi@JASC-ESCRITORIO:~/Angel/VirtualBox/Ubuntu/S.O/Bash$ ./bash
$ a
Comando inválido o no permitido.
$ printPpid
ID del proceso padre: 5567
$ printPid
ID del proceso padre: 5771
$ printGroup
ID del grupo del proceso: 5567
$ printSesion
ID del proceso: 5934
ID de la sesión: 5494
$ printUser
Nombre de usuario: lupi
Directorio Home: /home/lupi
UID: 1000
GID: 1000
$ printSerieFibonacci
Uso: printSerieFibonacci <numero>
El comando falló con el estado 1
$ printSerieFibonacci 5
Serie Fibonacci de 5 elementos:
0 1 1 2 3
$ exit
Comando inválido o no permitido.
$ EXIT
lupi@JASC-ESCRITORIO:~/Angel/VirtualBox/Ubuntu/S.O/Bash$