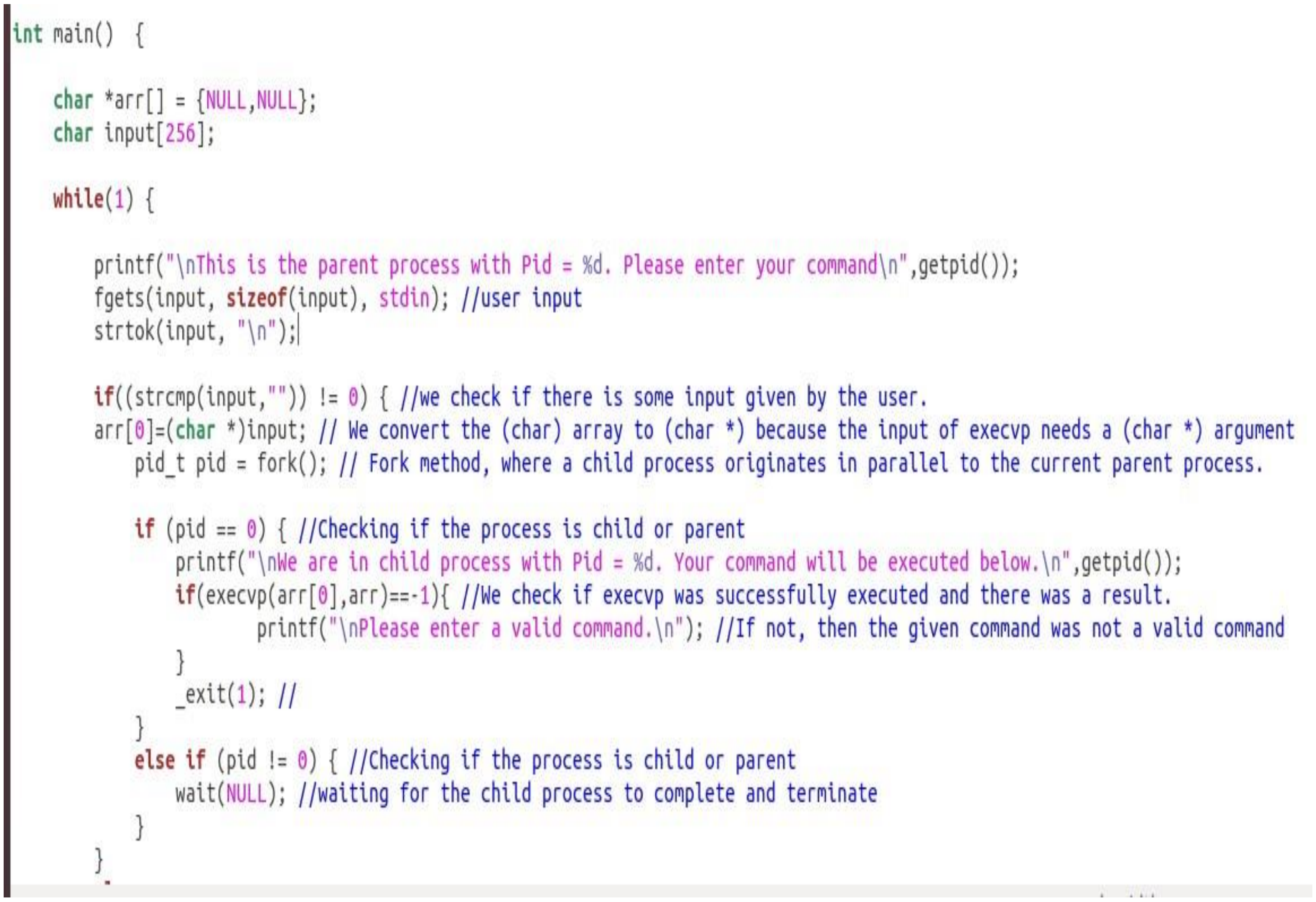
OS - Assignment 1 (Fork And Exec)

Net id – yhr208 (Yash Hitendra Rathod)

Problem: We have to write a c program that creates a parent process and asks the user for an input. The parent then creates a child process and this child then executes the input given by the user. After successfully executing, the child process terminates and the parent process then asks the user for more input.

We use the ***fork(), exec(), and wait()*** set of commands to achieve this.

We take the user input i.e the command which the user would like to execute eg. ls, find, date, etc



```
int main()  {

    char *arr[] = {NULL,NULL};
    char input[256];

    while(1) {

        printf("\nThis is the parent process with Pid = %d. Please enter your command\n",getpid());
        fgets(input, sizeof(input), stdin); //user input
        strtok(input, "\n");

        if((strcmp(input,"")) != 0) { //we check if there is some input given by the user.
        arr[0]=(char *)input; // We convert the (char) array to (char *) because the input of execvp needs a (char *) argument
            pid_t pid = fork(); // Fork method, where a child process originates in parallel to the current parent process.

            if (pid == 0) { //Checking if the process is child or parent
                printf("\nWe are in child process with Pid = %d. Your command will be executed below.\n",getpid());
                if(execvp(arr[0],arr)==-1){ //We check if execvp was successfully executed and there was a result.
                        printf("\nPlease enter a valid command.\n"); //If not, then the given command was not a valid command
                }
                _exit(1); //
            }
            else if (pid != 0) { //Checking if the process is child or parent
                wait(NULL); //waiting for the child process to complete and terminate
            }
        }
```

pid_t pid = fork() creates a parallel child process from the current parent process and the execution of this child starts after the fork() statement.

For the child process the pid is set to 0 by the fork(); and for the parent process the pid is the process Id of the child process.
We distinguish the two process with the value of the pid. i.e (0 for child; >0 for parent)
If pid<0 then there was an error during the child process creation and we return an error.

execvp(arr[0],arr)

We used execvp() to run the given input. The execvp returns integer -1 if the given command is not valid or if the execution fails.
We check the return value of the exec method. And if it returns -1, we output "Invalid command" else the results of the command execution is displayed.

We have also used the **wait()** system command. This is used to make the parent process wait till the child process is terminated.

After completion of the execvp(), and parent process again repeats the same process by asking the user for another input.