OS
LAB 1

Name: Lina Hazem Mohamed

ID: 5613

Simple Shell:

Main idea is to do a simple terminal as the terminal in Linux. It supports those commands:

1. Command “exit” to terminate the shell
2. Commands with no arguments (ls, cp, rm….)
3. Commands with arguments (ls -l)
4. Command executes in the background using & (firefox &)

Code:

-The following function reads the line from the user (storing the whole line until it finds “\n” which means a new line) and if it’s exit it will terminate the program, if not it will go to the function split into words.

```
int readLine(char* word, char line[])
{
   int i = 0, count = 0;
   fgets(line,100,stdin);
   //remove \n from line
   while(line[i] != '\n')
      i++;
   line[i] = '\0';
   if(strcmp(line,"exit") == 0)
      exit(0);
   count = split_into_words(word,line);
   return count; }
```

-The following function splits the line into words, every time space is found in the line we take the word before it and store it in an array and there is a counter to know how many words in the line.

```
int split_into_words(char* word[], char line[])
{
   int i = 0;
   char* ptr = strtok(line," ");
   while(ptr != NULL)
   {
      word[i++] = ptr;
      ptr = strtok(NULL," ");
   }
   return i;
}
```

-The following function is changing the directory you want to search in.

```
int cd(char *path)
{
   return chdir(path);
}
```

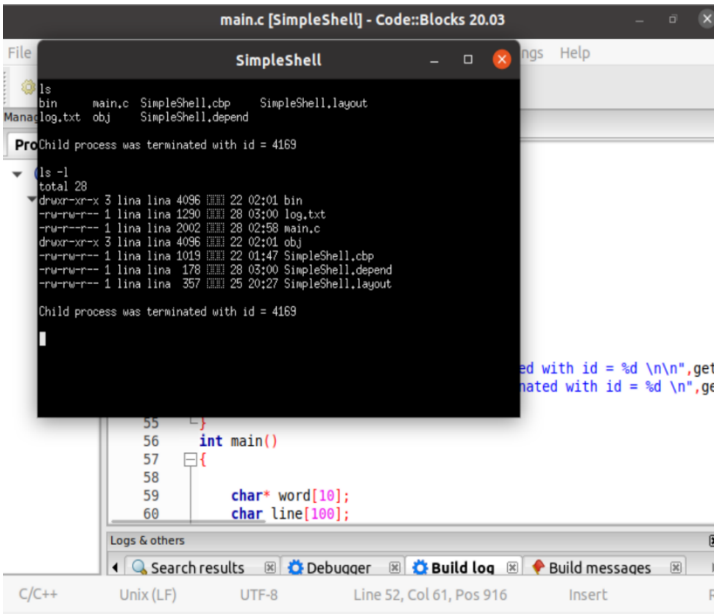
-The following function is writing in a file every time the child process is terminated with its ID.

```
void signalHandler(int sig)
{
   FILE* f;
   wait(NULL);
   f = fopen("log.txt","a");
   printf("\nChild process was terminated with id = %d \n",getpid());
   fprintf(f,"\nChild process was terminated with id = %d \n",getpid());
   fclose(f);
}
```

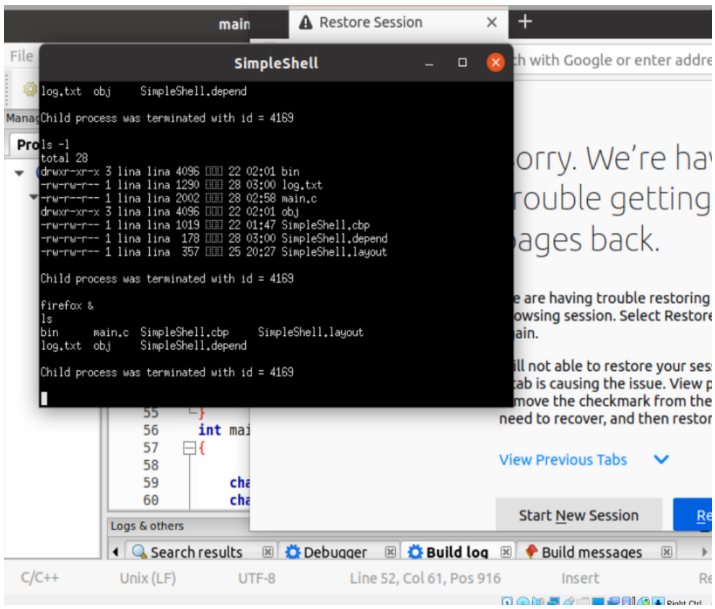
-The following function is the main function, the call to signal takes a pointer to function signalHandler and the SIGCHLD signal is sent to the parent of a child process when it terminates. And start an infinite loop, read the line which returns the number of words in “count”. The fork function creates a new process by duplicating the existing process and pid is the id of the child process. Compare the last word if it’s “&” then make the variable background = 1 and make the last word has no value by NULL. Compare the first word if equals “cd” change directory. If fork successes and in the child process execute the commands like (ls, pwd, ls -l….), if it’s in the parent process there are 2 options: if variable background = 1 then we are in the background process and there is an & in the command so continue and don’t wait for the child process to terminate, if not then we will wait till the child process terminates.

```
int main()
{
   char* word[10];
   char line[100];
   int count, background = 0;
   signal(SIGCHLD,signalHandler);
   while(1)
   {
      count = readLine(word,line);
```

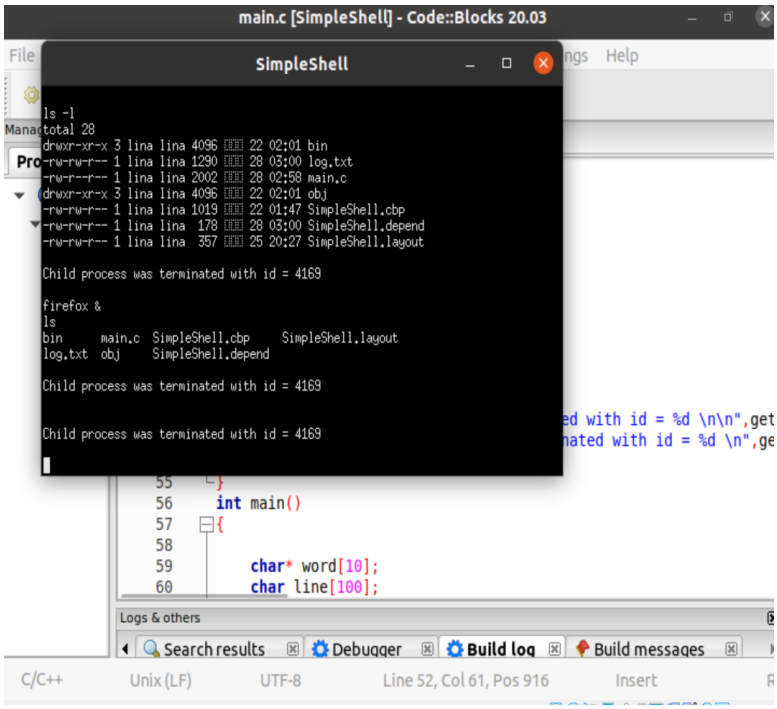
```
        pid_t pid = fork();

        if(strcmp(word[count-1],"&") == 0)
        {
           background = 1;
           word[count-1] = NULL;
        }
        if(strcmp(word[0],"cd") == 0)
        {
           if(cd(word[1])<0)
              perror(word[1]);
           continue;   /* skip fork*/
        }
        if(pid >= 0)  // fork success
        {
           if(pid == 0)  //child proccess
           {
              if(execvp(word[0],word)<0)
                 perror(word[0]);
           }
           else    //parent process
           {
              if(background == 1)
                 continue;
              waitpid(pid,NULL,0);  // wait until child process finishes
           }
        }
        else
           printf("Fork Failed\n");
     }
     return 0;
}
```



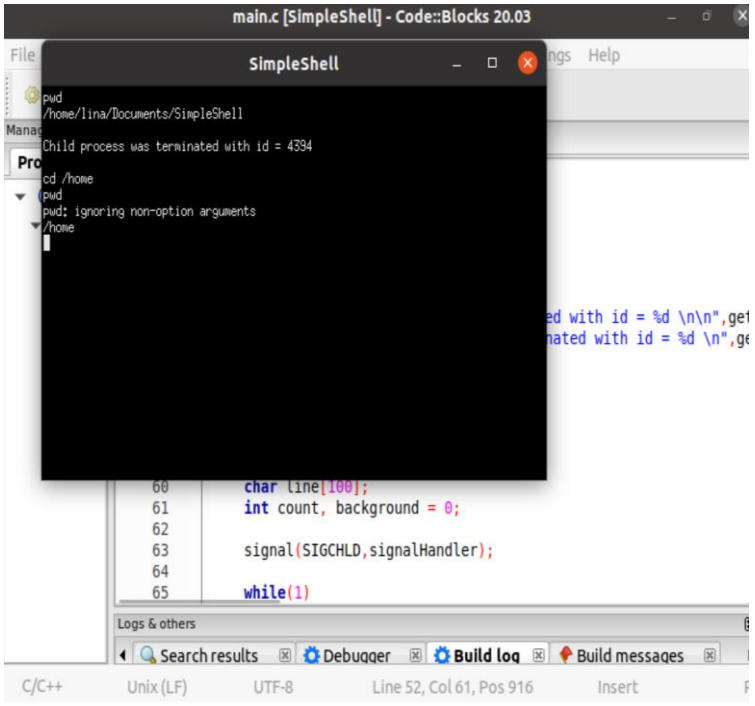
The commands with or without arguments



Doing 2 processes at the same time



After closing firefox this process is terminated



Changing the directory