Assignment-4
Fork assignment-3

Subash Mylraj
(CED18I051)

16 September 2020

Question 3: Develop a C program to count the maximum number of processes that can be created using fork call.

Code:

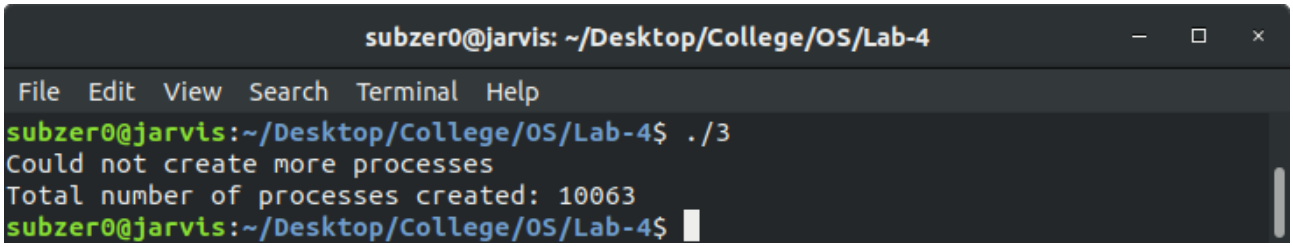
```c
#include<stdio.h>
#include<sys/wait.h>
#include<unistd.h>
#include<stdlib.h>

int main (){

   int count = 0;

   while(1){
      pid_t pid = fork();
      if(pid == 0){
         sleep(15);
         exit(0);
      }
      count += 1;
      if(pid < 0){
         printf("Could not create more processes\n");
         break;
      }
   }

   printf("Total number of processes created: %d\n", count);
   wait(NULL);

   return 0;
}
```

Explanation:

This code creates child processes in an infinite loop till it reaches a point when it cannot create anymore processes. This limit can be assumed as the cap on the number of processes that can be created.

To avoid the issue of child processes calling fork repeatedly, the child processes are made to sleep for a time of 10 seconds. So essentially, this program finds the total number of processes that can be created in about 10 seconds (its a bit more than 10 seconds as there is time taken for context switching and other statements). Increasing the sleep time for each process, produced similar results.

Output:



Question 4: Develop your own command shell [say mark it with @] that accepts user commands (System or User Binaries), executes the commands and returns the prompt for further user interaction. Also extend this to support a history feature (if the user types !6 at the command prompt; it shud display the most recent execute 6 commands). You may provide validation features such as !10 when there are only 9 files to display the entire history contents and other validations required for the history feature;

Code:

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

struct list{
   int count;
   struct node *head;
};

struct node{
   char *command;
   struct node *next;
};

struct list history;

void insert (char *);
void show (int);
void get_cmd(char *[]);

int main (){

   char hostname[HOST_NAME_MAX], username[LOGIN_NAME_MAX], work_dir[PATH_MAX];
   gethostname(hostname, 20);
   getlogin_r(username, 20);

   history.count = 0;
   history.head = NULL;
```

```
   printf("\033[1;31m-------------Welcome to the new terminal-------------\033[0m\n");
   char *argv[100];

   while(1){

      getcwd(work_dir, sizeof(work_dir));
      printf("\033[1;36m%s@\033[1;35m%s:\033[01;33m%s\033[1;31m@ \033[0m", username, hostname,
          work_dir);
      // printf("\033[1;36msubash:\033[0m ");
      char *argv[100];
      get_cmd(argv);

      if(strcmp(argv[0],"exit")==0){
         exit(0);
      }
      else if(argv[0][0] == '!'){
         if(argv[1] != NULL){
            // printf("%s\n", argv[0][1]);
            show(atoi(argv[1]));
         }
         else{
            // printf("%s\n", argv[0][2]);
            show(10);
         }
      }
      else if(strcmp(argv[0],"cd")==0){
         chdir(argv[1]);
      }
      else{
         pid_t pid = vfork();
         if(pid == 0){
            execvp(argv[0], argv);
         }
         else{
            wait(NULL);
         }
      }
   }
   return 0;
}

void get_cmd(char *argv[100]){

   char *user_input = (char *)malloc(100);
   scanf(" %[^\n]", user_input);

   char *cmd = (char *)malloc(100);
   strcpy(cmd, user_input);
   insert(cmd);

   argv[0] = strtok(user_input, " ");
   int i=0;

   while (argv[i] != NULL) {
      i++;
      argv[i] = strtok(NULL, " ");
   }
}

void insert (char *cmd){

   struct node* temp = (struct node*)malloc(sizeof(struct node));
   // strcmp(cmd, '\0');
   temp->command = cmd;
   // strcpy(temp->command, cmd);
```

```
   if(history.head == NULL){
      temp->next = NULL;
      history.head = temp;
      history.count = 1;
      return;
   }

   temp->next = history.head;
   history.head = temp;

   if(history.count == 10){
      // printf("\n%s:\n", cmd);
      struct node* temp = history.head, temp1;

      for(int i=0; i<9; i++){
         temp = temp->next;
         // printf("test\n");
      }
      temp->next = NULL;
      // show();
   }
   else{
      history.count += 1;
   }

}

void show (int n){

   // printf("count = %d\n", n);
   struct node* temp = history.head;
   int counter = 0;
   while(temp != NULL && counter < n){
      printf("%s\n", temp->command);
      temp = temp->next;
      counter += 1;
   }
}
```

Explanation:

This code creates a new terminal interface. Any command that is located in the */bin/* directory will work. Sadly piping logics do not work on this terminal. The command *exit* can be used to exit the terminal. Though it is to be noted that, this command is not called through exec statements.

History of the last 10 commands can be displayed by typing !. Further, typing !5 shows the last 5 commands used. Typing a value more than 10 will only print the last 10 commands used.

Colours were added using \033 in printf statements.

Directories can be changed by using the cd command. This command is not called through exec calls rather by calling the chdir functions. The prompt displays the current username and hostname using *gethostname*() and *getlogin_r*() functions. The current working directory information is obtained by using the *getcwd*() functinon.

Output:

```
subzer0@jarvis: ~/Desktop/College/OS/Lab-4
File  Edit  View  Search  Terminal  Help
subzer0@jarvis:~/Desktop/College/OS/Lab-4$ ./4
-------------Welcome to the new terminal-------------
subzer0@jarvis:/home/subzer0/Desktop/College/OS/Lab-4@ cd ..
subzer0@jarvis:/home/subzer0/Desktop/College/OS@ cd ../..
subzer0@jarvis:/home/subzer0/Desktop@ ls
 Cocounter
 College
 labelImg
 meme
'Mia & Sebastian'\''s Theme (from La La Land) [from pianounchained.com].pdf'
subzer0@jarvis:/home/subzer0/Desktop@ ls -a
 .
 ..
 Cocounter
 College
 labelImg
 meme
'Mia & Sebastian'\''s Theme (from La La Land) [from pianounchained.com].pdf'
subzer0@jarvis:/home/subzer0/Desktop@ !
!
ls -a
ls
cd ../..
cd ..
subzer0@jarvis:/home/subzer0/Desktop@ ! 2
! 2
!
subzer0@jarvis:/home/subzer0/Desktop@ ! 4
! 4
! 2
!
ls -a
subzer0@jarvis:/home/subzer0/Desktop@ exit
subzer0@jarvis:~/Desktop/College/OS/Lab-4$
```