Name : Hassan Ghazy Sammour

ID : 120170878

Project Shell

ENG : Mohammed Nafiz Almadhoun

Date: 12/23/2020

The shell must support the following internal commands:

i. cd <directory>—Change the current default directory to <directory>. If the <directory> argument is not present, report the current directory. If the directory does not exist, an appropriate error should be reported. This command should also change the PWD environment variable. ✔

ii. clr—Clear the screen. ✔

iii. dir <directory>—List the contents of directory <directory>. ✔

iv. environ—List all the environment strings. ✔

v. echo <comment>—Display <comment> on the display followed by a new line (multiple spaces/tabs may be reduced to a single space). ✔

vi. help—Display the user manual using the more filter. ✔

vii. pause—Pause operation of the shell until “Enter” is pressed. ✔

viii. quit—Quit the shell. ✔

ix. The shell environment should contain shell=<pathname>/myshell where <pathname>/myshell is the full path for the shell executable (not a hardwired path back to your directory, but the one from which it was executed). ✔

```
hassan@hassan-VirtualBox: ~/project
assan@hassan-VirtualBox:~/project$ gcc project.c -o out
assan@hassan-VirtualBox:~/project$ ./out
home/hassan/project/myshell$ dir
ut  project.c
home/hassan/project/myshell$ ls
ut  project.c
home/hassan/project/myshell$ echo cimment
imment
home/hassan/project/myshell$ pause
ress ENTER key to Can write your commands
eow meow
home/hassan/project/myshell$ ls *a
s: cannot access '*a': No such file or directory
home/hassan/project/myshell$ ls -a
  ..  out  project.c
home/hassan/project/myshell$ env
HELL=/bin/bash
ESSION_MANAGER=local/hassan-VirtualBox:@/tmp/.ICE-unix/1244,unix/hassan-Virtua
Box:/tmp/.ICE-unix/1244
T_ACCESSIBILITY=1
OLORTERM=truecolor
DG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
DG_MENU_PREFIX=gnome-
NOME_DESKTOP_SESSION_ID=this-is-deprecated
TK_IM_MODULE=ibus
T4_IM_MODULE=ibus
NOME_SHELL_SESSION_MODE=ubuntu
SH_AUTH_SOCK=/run/user/1000/keyring/ssh
MODIFIERS=@im=ibus
```

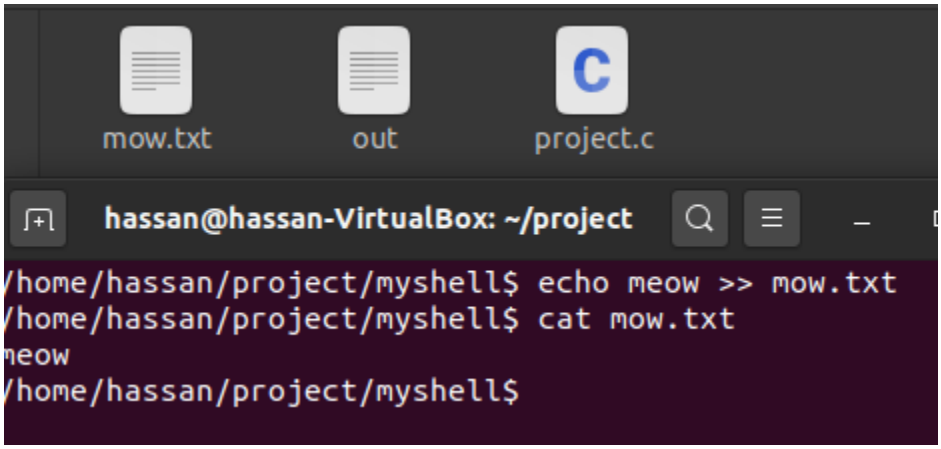
After clear

```
hassan@hassan-VirtualBox: ~/project
/home/hassan/project/myshell$
```

2. All other command line input is interpreted as program invocation, which
should be done by the shell forking and execing the programs as its own child
processes. The programs should be executed with an environment that contains
the entry: parent=<pathname>/myshell where <pathname>/myshell
is as described in 1.ix above. ✔

3. The shell must be able to take its command line input from a file. That is, if the
shell is invoked with a command line argument:
myshell batchfile
then batchfile is assumed to contain a set of command lines for the shell to
process. When the end-of-file is reached, the shell should exit. Obviously, if the
shell is invoked without a command line argument, it solicits input from the user
via a prompt on the display. ✕

4. The shell must support I/O redirection on either or both stdin and/or stdout.
That is, the command line
programname arg1 arg2 < inputfile > outputfile
will execute the program programname with arguments arg1 and arg2, the
stdin FILE stream replaced by inputfile and the stdout FILE stream replaced by outputfile.
stdout redirection should also be possible for the internal commands dir,
environ, echo, and help.
With output redirection, if the redirection character is > then the outputfile is created if it does not exist, and truncated if it does. If the redirection token is >> then outputfile is created if it does not exist, and appended to if it does. ✔



5. The shell must support background execution of programs. An ampersand (&)
at the end of the command line indicates that the shell should return to the command line prompt immediately after launching that program. ✕

6. The command line prompt must contain the pathname of the current directory. ✔

This is snippet from the original code:

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

int pipe_count=0, fd;
static char* args[512];
char input_buffer[1024];
char *cmd_exec[100];
int flag, len;
char cwd[1024];
pid_t pid;
int environment_flag;
int output_redirection, input_redirection;
int pid, status;
char history_data[1000][1000];
char current_directory[1000];
char *input_redirection_file;
char *output_redirection_file;
extern char** environ;
static int command(int, int, int, char *cmd_exec);

void sigintHandler(int sig_num)
```

C Tab Width: 8 Ln 27, Col 1 INS

```c
void sigintHandler(int sig_num)
{
    signal(SIGINT, sigintHandler);
    fflush(stdout);
}
void clear()
{
  fd =0;
  flag=0;
  len=0;
  pipe_count=0;
  output_redirection=0;
  input_redirection=0;
  input_buffer[0]='\0';
  cwd[0] = '\0';
  pid=0;
  environment_flag=0;
}
```

```c
void environmment()
{
  int i =1, index=0;
  char env_val[1000], *value;
  while(args[1][i]!='\0')
              {
                   env_val[index]=args[1][i];
                   index++;
                    i++;
              }
  env_val[index]='\0';
  value=getenv(env_val);

  if(!value)
      printf("\n");
  else printf("%s\n", value);
}


void change_directory()
{
char *h="/home";
if(args[1]==NULL)
        chdir(h);
else if ((strcmp(args[1], "~")==0) || (strcmp(args[1], "~/")==0))
        chdir(h);
else if(chdir(args[1])<0)
    printf("bash: cd: %s: No such file or directory\n", args[1]);
```

C ▾ Tab Width: 8 ▾ Ln 27, Col 1 ▾

```c
else if(chdir(args[1])<0)
    printf("bash: cd: %s: No such file or directory\n", args[1]);

}

void parent_directory()
{
if (getcwd(cwd, sizeof(cwd)) != NULL)
        {
         printf("%s\n", cwd );
        }
else
       perror("getcwd() error");


}
void echo_calling(char *echo_val)
{
  int i=0, index=0;
  environmment_flag=0;
  char new_args[1024],env_val[1000], *str[10];
  str[0]=strtok(echo_val," ");
  str[1]=strtok(NULL, "");
  strcpy(env_val, args[1]);
  if(str[1]==NULL)
         {
```