

# CSE231: Operating Systems

## Assignment 1 Shell

**Name:** Vansh  
**Roll no.:** 2021363  
**Section:** B  
**Branch:** CSD

---

### Description

My shell named “HR36” is a simple unix shell which provides the user a command line interface to type commands in. In the code file “Shell.c” lies the main function and in which implementation of external commands through `fork()` & `wait()` system calls {in different functions present in different files} and through POSIX threads & `system()` {present in the main function itself} takes place.

### Assumptions

- Length of a single command cannot be more than 1024 characters.
- Only one command can run at a time. Eg- “cd .. ” and “cd \* ” can be executed but “cd .. \*” cannot be executed.

### Commands Handled

#### cd

##### *Options*

- `cd` or `cd *` //for going back to the home directory
- `cd ..` //for going back to the previous directory
- `cd #name of the directory` // for going into the specified directory

##### *Errors Handled*

- “No such file or directory” //if user enters a non-existing directory
- “Too many arguments” //if user enters more than one directory

## echo

### *Options*

- **echo \*** //prints all the files in the present directory
- **echo -n** //does not print the new trailing line

### *Errors Handled*

- echo itself does not require to throw any errors.

## pwd

### *Options*

- **pwd -L** //resolves all symbolic links
- **pwd -P** //avoids all symbolic links

### *Errors Handled*

- **"Unrecognized Option"** //if user inputs another option that isn't handled
- **"Too many arguments"** //if user enters more than one directory

## exit

### *Options*

No options required

### *Errors Handled*

- exit requires no error handling.

## ls

### *Options*

- **ls -r** //prints the list of files in reverse order
- **ls -n** //prints Group ID and Owner ID instead of their names

### *Errors Handled*

- **"Unrecognized Option"** //if user inputs another option that is not handled
- **"Too many arguments"** //if user enters more than one directory
- **"Invalid Option"** //if user provides an invalid option

## cat

### *Options*

- `cat -e` //to display \$ at the end of each line
- `cat -n` //to display line numbers in front of each line

### *Errors Handled*

- “No such file or directory” //if user enters a non-existing directory
- “Too many arguments” //if user enters more than one directory
- “Invalid Option” //if user provides an invalid option

## mkdir

### *Options*

- `mkdir -v` //prints name of the directory after creation
- `mkdir -p` //can make a directory inside a directory

### *Errors Handled*

- “Unrecognized Option” //if user inputs another option that is not handled
- “Too many arguments” //if user enters more than one directory
- “Invalid Option” //if user provides an invalid option

## rm

### *Options*

- `rm -rf` //deletes a directory without reading files in it
- `rm -d` //Only deletes an empty directory and throws a message if directory is not empty

### *Errors Handled*

- “Unrecognized Option” //if user inputs another option that is not handled
- “Too many arguments” //if user enters more than one directory
- “Invalid Option” //if user provides an invalid option

## date

### *Options*

- `date -u` //prints Coordinated Universal Time (UTC)

- **date -r** //prints date and time of last modification of file

### *Errors Handled*

- **“Unrecognized Option”** //if user inputs another option that is not handled
- **“Too many arguments”** //if user enters more than one directory
- **“Invalid Option”** //if user provides an invalid option

## Implementing shell using POSIX threads

- **pthread\_create()** creates a new thread within a process, with attributes defined by the thread attribute object, attr, that is created by pthread\_attr\_init().
- **pthread\_t** is the data type used to uniquely identify a thread. It is returned by pthread\_create().
- **pthread\_join()** allows the calling thread to wait for the ending of the target thread.
- **pthread\_exit()** ends the calling thread and makes status available to any thread that calls pthread\_join() with the ending thread's thread ID.
- **system()** calls the system linux commands by taking user input as an argument.
- Shell implementation by pthreads will work when **“&t”** is followed by any external command.

## Test cases

\*The test cases will run as desired when initially the code folder is on the desktop.

### Test case for shell implementation only through system calls(for external commands)

```
{
> $ cd ..
> $ cd ..
> $ cd Desktop
> $ mkdir -v final
> $ cd final
> $ cd *
> $ cd * ..
> $ pwd -L
> $ cd Desktop
> $ ls -r
> $ ls -n
> $ ls-r
> $ cat -e U.txt
> $ cat -n U.txt
```

```

> $ cat -e S.txt
> $ mkdir -p final/final1
> $ pwd -L
> $ cd final
> $ rm -d final1
> $ cd ..
> $ date -u
> $ echo *
> $ date -r U.txt
> $ echo -n "vansh"
> $ cat
> $ vansh
> $ yadav
> $ ^C
> $ exit
}

```

**Test case for shell implementation only through POSIX threads(for external commands)**

```

{
> $ cd ..
> $ cd ..
> $ cd Desktop
> $ &t mkdir -v final
> $ cd final
> $ cd *
> $ cd * ..
> $ pwd -L
> $ cd Desktop
> $ &t ls -r
> $ &t ls -n
> $ &t ls-r
> $ &t cat -e U.txt
> $ &t cat -n U.txt
> $ &t cat -e S.txt
> $ &t mkdir -p final/final1
> $ pwd -L
> $ cd final
> $ &t rm -d final1
> $ cd ..
> $ &t date -u
> $ echo *
> $ &t date -r U.txt
> $ echo -n "vansh"
> $ &t cat
> $ vansh
> $ yadav

```

```
> $ ^C  
> $ exit  
}
```