Basic Linux/Unix Shell

System Description:

The shell a software which is used to execute commands which are used to communicate with the system. It is mostly used execute operations/instructions on files and directories. We pass in commands and their arguments and internally the system calls the 'system()' function to execute them. It creates threads/forks in order to execute the commands while running the main shell.
In this Assignment, we have execute commands using fork()/exec() as well as threads()/system() families of functions.

Assumptions:

1. There are only two flags which are implemented per command.
2. The flags must be specified immediately after the command, e.g., 'rm -i filename' is legal however 'rm filename -i' is not legal.
3. Home directory is '/home/js' .
4. Printing files in column format along with their file numbers (arbitrary) is allowed.
5. Special character interpretation of strings in echo is disabled.
6. There are no flags for pwd.
7. Blank spaces ' ' are treated as arguments to provide protection and accuracy, therefore output may change if many unnecessary blank spaces are provided.
8. Two flags cannot be used simultaneously.
9. 'ls' does not print out directories. 'ls -a' can be used to view all files including directories.
10. Everything after 'date' in the date command can be treated as a flag.
11. Using unnecessary special characters is not allowed. Also putting forward slashes at the end of a command is not allowed, i.e. like, dir/file/, rather dir/file .

In general, an instruction will look like the following:
<command/command&t> [-flag] [arguments]

Basic Linux/Unix Shell

Note: In order to display that the file is being called through thread/fork, I have printed statements that indicate how the program is being executed before the execution of the program begins.

Note: The file path must be respecified in the ‘shell.c’ file in order for exec() and system() to enable execution of the command file.

Note: Commands can handle multiple arguments however care must be taken to not exceed the malloced memory (around 100).

1. echo

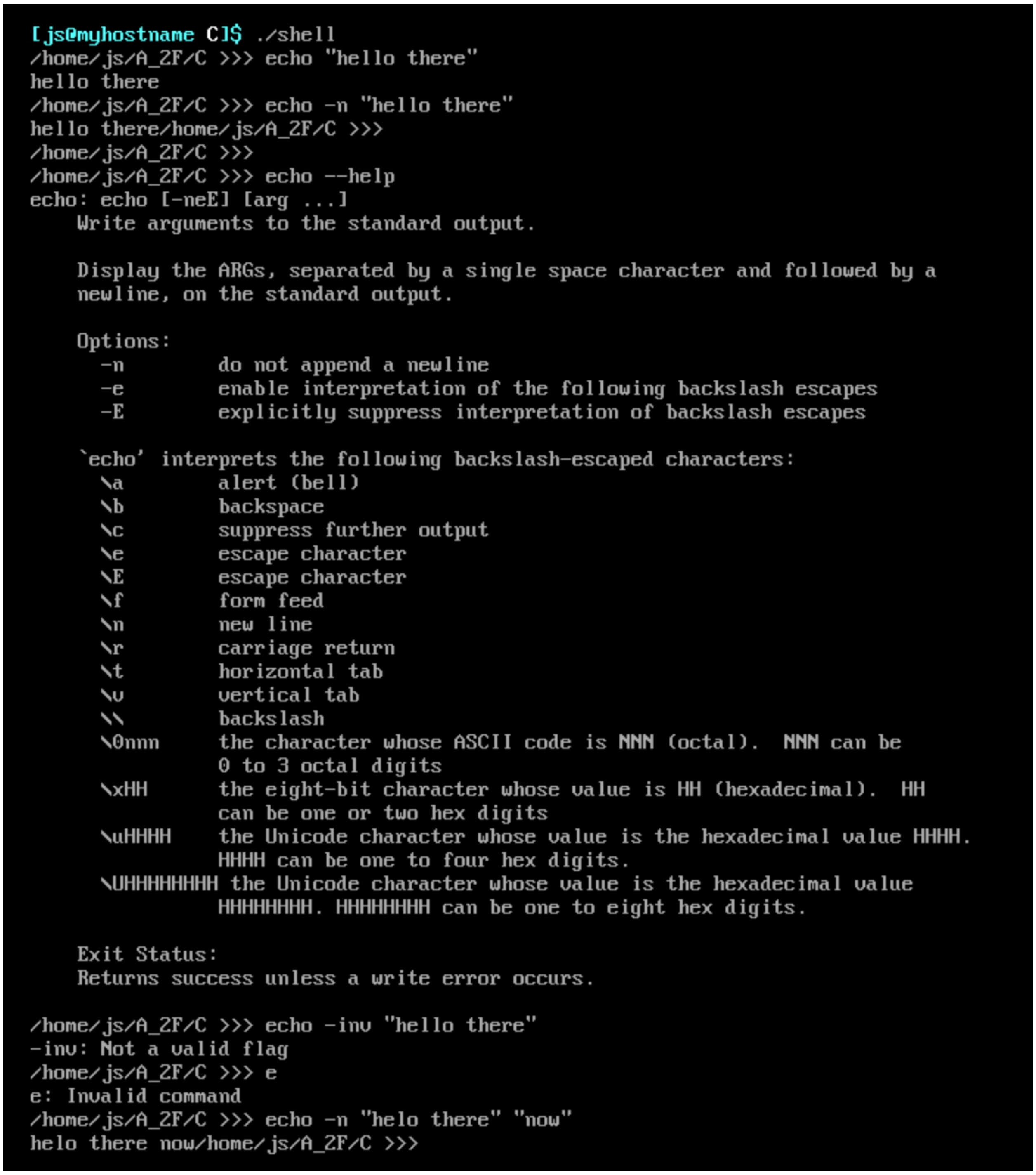
Flags handled:

1. -n : does not print a newline.
2. –-help : prints out the help page for echo.

Edge Cases:

1. echo “<string>”
   Doesn’t print out the double quotes.
2. echo -invalid_flag
   Prints out that the flag is invalid along with the flag.
3. echo “string” “string1”
   Prints out both the strings.

Basic Linux/Unix Shell



```
[js@myhostname C]$ ./shell
/home/js/A_2F/C >>> echo "hello there"
hello there
/home/js/A_2F/C >>> echo -n "hello there"
hello there/home/js/A_2F/C >>>
/home/js/A_2F/C >>>
/home/js/A_2F/C >>> echo --help
echo: echo [-neE] [arg ...]
    Write arguments to the standard output.

    Display the ARGs, separated by a single space character and followed by a
    newline, on the standard output.

    Options:
      -n        do not append a newline
      -e        enable interpretation of the following backslash escapes
      -E        explicitly suppress interpretation of backslash escapes

    `echo' interprets the following backslash-escaped characters:
      \a        alert (bell)
      \b        backspace
      \c        suppress further output
      \e        escape character
      \E        escape character
      \f        form feed
      \n        new line
      \r        carriage return
      \t        horizontal tab
      \v        vertical tab
      \\        backslash
      \0nnn     the character whose ASCII code is NNN (octal).  NNN can be
                0 to 3 octal digits
      \xHH      the eight-bit character whose value is HH (hexadecimal).  HH
                can be one or two hex digits
      \uHHHH    the Unicode character whose value is the hexadecimal value HHHH.
                HHHH can be one to four hex digits.
      \UHHHHHHHH the Unicode character whose value is the hexadecimal value
                HHHHHHHH. HHHHHHHH can be one to eight hex digits.

    Exit Status:
    Returns success unless a write error occurs.

/home/js/A_2F/C >>> echo -inv "hello there"
-inv: Not a valid flag
/home/js/A_2F/C >>> e
e: Invalid command
/home/js/A_2F/C >>> echo -n "helo there" "now"
helo there now/home/js/A_2F/C >>>
```

Basic Linux/Unix Shell

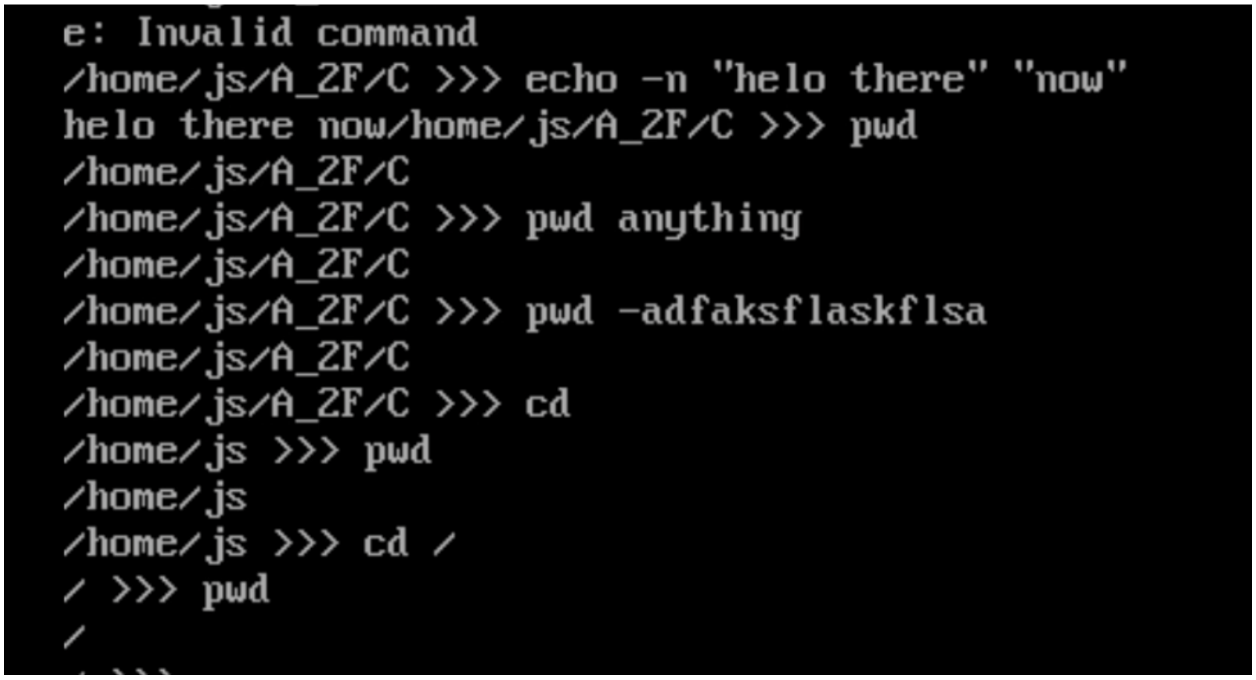
2. pwd

Flags handled: We were told not to handle any flags for pwd.

Edge Cases:

1. pwd any_random_string
   Prints the current working directory regardless of what is passed as an argument.

2. If we change the directory, pwd works properly.



```
e: Invalid command
/home/js/A_2F/C >>> echo -n "helo there" "now"
helo there now/home/js/A_2F/C >>> pwd
/home/js/A_2F/C
/home/js/A_2F/C >>> pwd anything
/home/js/A_2F/C
/home/js/A_2F/C >>> pwd -adfaksflaskflsa
/home/js/A_2F/C
/home/js/A_2F/C >>> cd
/home/js >>> pwd
/home/js
/home/js >>> cd /
/ >>> pwd
/
```

Basic Linux/Unix Shell

3. cd

Flags handled:

.. : changes the working directory to the parent directory of the current directory.

~ : changes the working directory to the home directory.

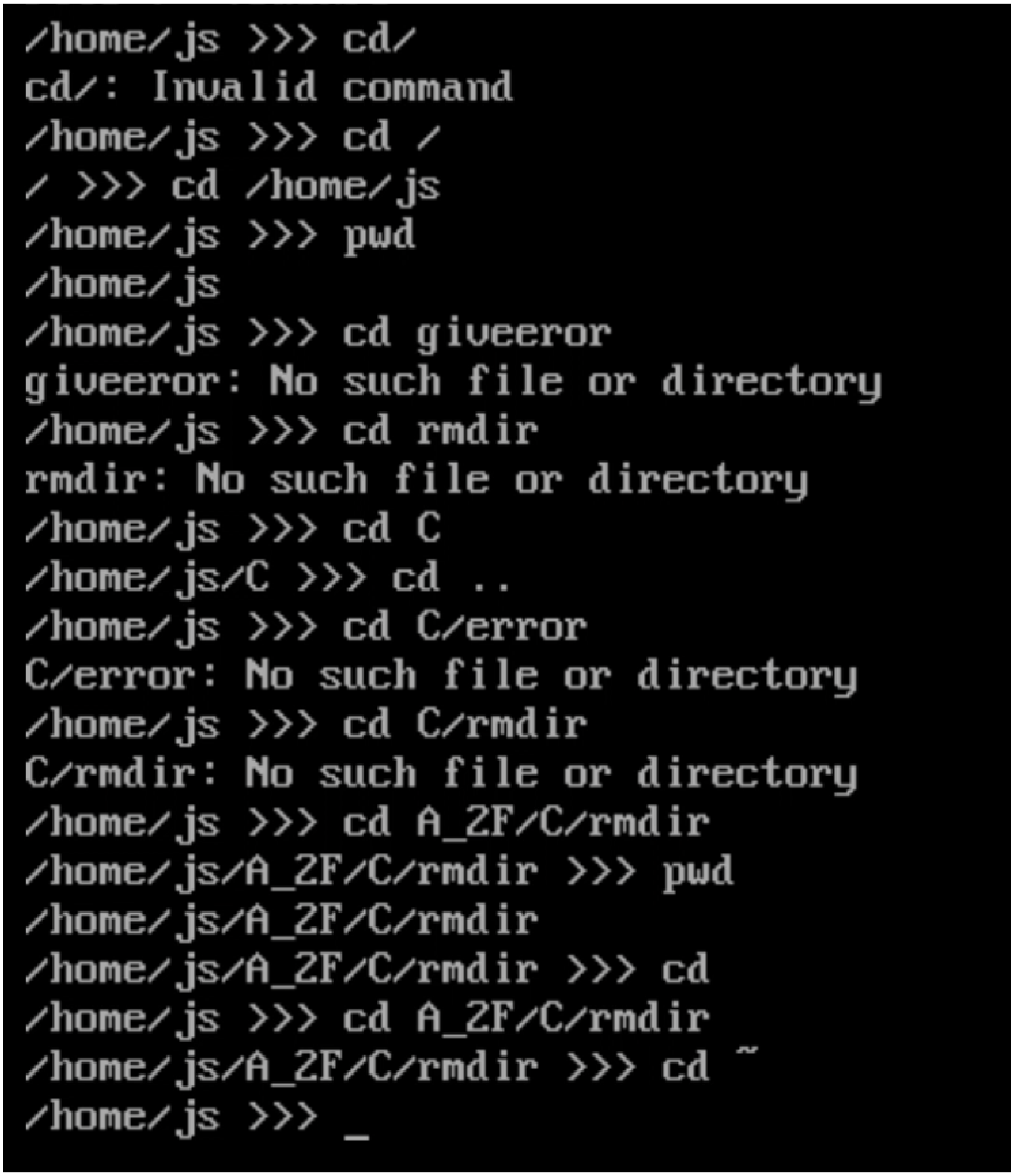
/ : changes the working directory to the root directory.

Edge Cases:

1. cd
   changes the working directory to the home directory.

2. cd invalid_location
   Prints out “No such file or directory” along with the directory name that is not found.

Basic Linux/Unix Shell



```
/home/js >>> cd/
cd/: Invalid command
/home/js >>> cd /
/ >>> cd /home/js
/home/js >>> pwd
/home/js
/home/js >>> cd giveeror
giveeror: No such file or directory
/home/js >>> cd rmdir
rmdir: No such file or directory
/home/js >>> cd C
/home/js/C >>> cd ..
/home/js >>> cd C/error
C/error: No such file or directory
/home/js >>> cd C/rmdir
C/rmdir: No such file or directory
/home/js >>> cd A_2F/C/rmdir
/home/js/A_2F/C/rmdir >>> pwd
/home/js/A_2F/C/rmdir
/home/js/A_2F/C/rmdir >>> cd
/home/js >>> cd A_2F/C/rmdir
/home/js/A_2F/C/rmdir >>> cd ~
/home/js >>> _
```

Basic Linux/Unix Shell

4. ls

Flags handled:

1. -a : prints out all the files present in the directory.

2. -r : prints out the files present in the directory in reverse order.

Edge Cases:

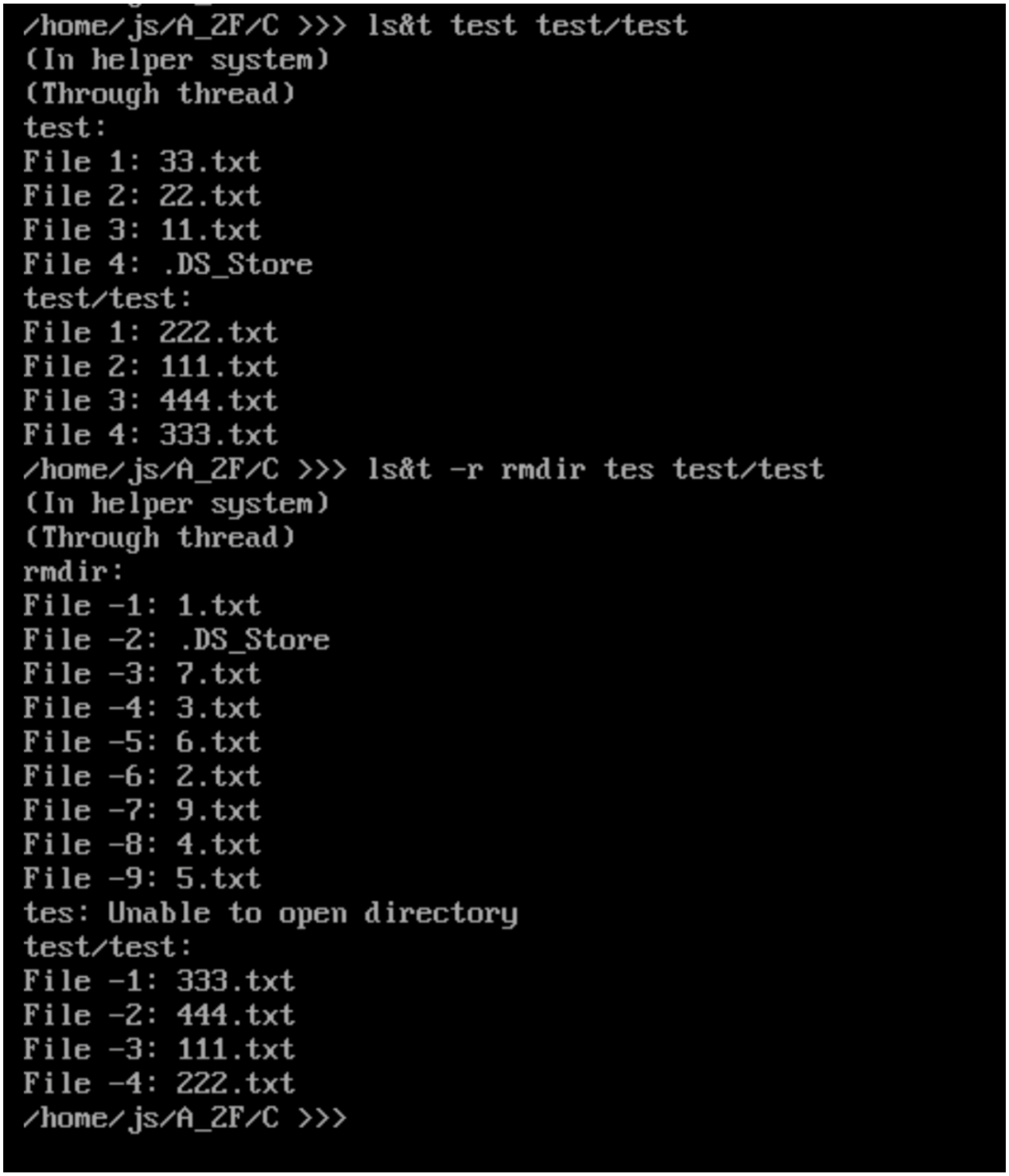
1. ls invalid_location
   Prints out “Invalid location” along with the directory name that is invalid but works for the valid locations.

2. ls -invalid_flag
   Prints out “Invalid command” along with the flag that is invalid.

3. ls
   Prints out all the (regular) files present in the current working directory.

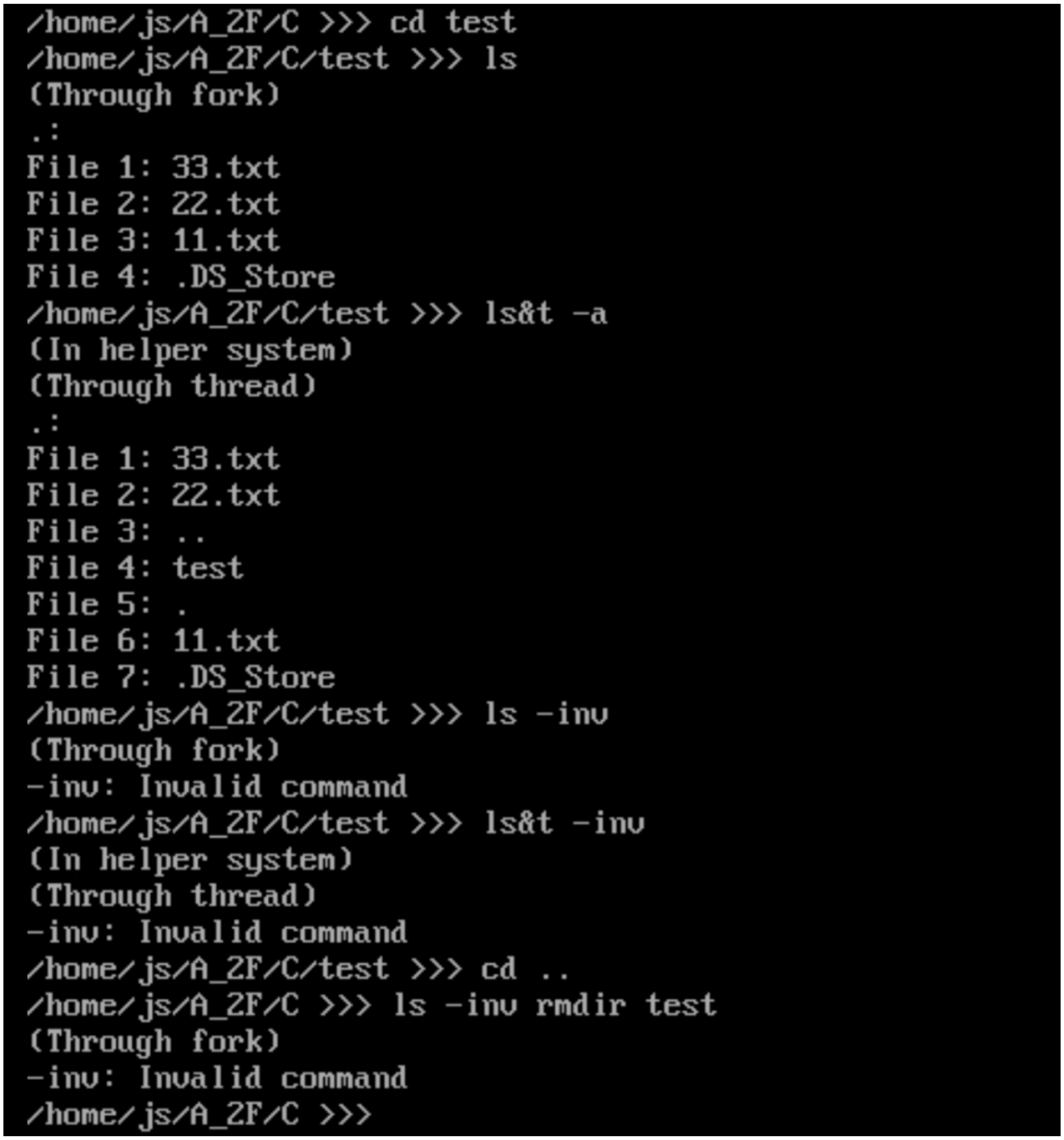
4. ls filename
   Prints out “Invalid location” as a file is not openable.

Basic Linux/Unix Shell



```
/home/js/A_2F/C >>> ls&t test test/test
(In helper system)
(Through thread)
test:
File 1: 33.txt
File 2: 22.txt
File 3: 11.txt
File 4: .DS_Store
test/test:
File 1: 222.txt
File 2: 111.txt
File 3: 444.txt
File 4: 333.txt
/home/js/A_2F/C >>> ls&t -r rmdir tes test/test
(In helper system)
(Through thread)
rmdir:
File -1: 1.txt
File -2: .DS_Store
File -3: 7.txt
File -4: 3.txt
File -5: 6.txt
File -6: 2.txt
File -7: 9.txt
File -8: 4.txt
File -9: 5.txt
tes: Unable to open directory
test/test:
File -1: 333.txt
File -2: 444.txt
File -3: 111.txt
File -4: 222.txt
/home/js/A_2F/C >>>
```

Basic Linux/Unix Shell



```
/home/js/A_2F/C >>> cd test
/home/js/A_2F/C/test >>> ls
(Through fork)
.:
File 1: 33.txt
File 2: 22.txt
File 3: 11.txt
File 4: .DS_Store
/home/js/A_2F/C/test >>> ls&t -a
(In helper system)
(Through thread)
.:
File 1: 33.txt
File 2: 22.txt
File 3: ..
File 4: test
File 5: .
File 6: 11.txt
File 7: .DS_Store
/home/js/A_2F/C/test >>> ls -inv
(Through fork)
-inv: Invalid command
/home/js/A_2F/C/test >>> ls&t -inv
(In helper system)
(Through thread)
-inv: Invalid command
/home/js/A_2F/C/test >>> cd ..
/home/js/A_2F/C >>> ls -inv rmdir test
(Through fork)
-inv: Invalid command
/home/js/A_2F/C >>>
```

Basic Linux/Unix Shell

5. date

Flags handled:

1. -u : prints out the UTC Time

2. -I : prints out the date in Indian Standard format, that is, in YYYY/MM/DD

Edge Cases:

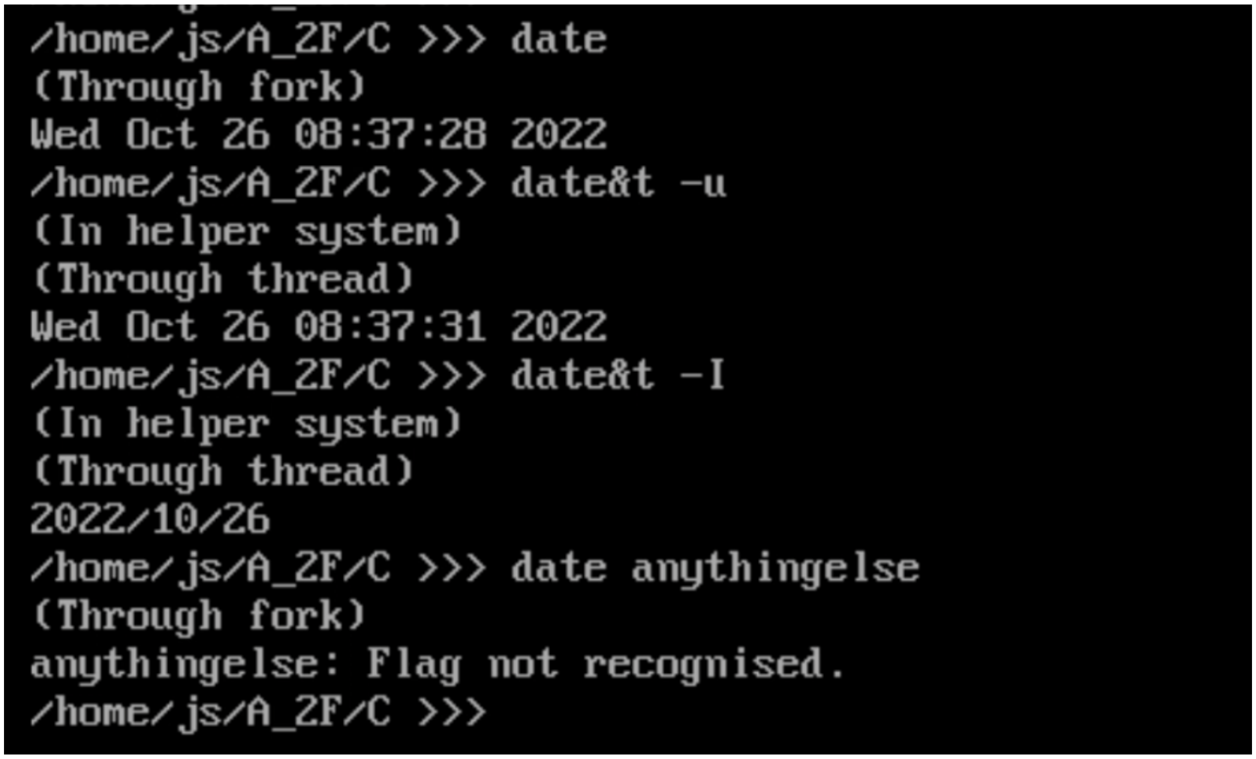
1. date -invalid_flag
   Prints out “Invalid flag” along with the flag that is invalid.

2. date any_non_flag_string
   Prints out “Invalid flag” as it takes in only a flag and no arguments (that is, everything afer the command ‘date’ is treated as a flag).

3. date
   When no argument is provided, it prints the date-time of the system.

Note: Date time of the system is set to UTC.

Basic Linux/Unix Shell



```
/home/js/A_2F/C >>> date
(Through fork)
Wed Oct 26 08:37:28 2022
/home/js/A_2F/C >>> date&t -u
(In helper system)
(Through thread)
Wed Oct 26 08:37:31 2022
/home/js/A_2F/C >>> date&t -I
(In helper system)
(Through thread)
2022/10/26
/home/js/A_2F/C >>> date anythingelse
(Through fork)
anythingelse: Flag not recognised.
/home/js/A_2F/C >>>
```

Basic Linux/Unix Shell

6. cat

Flags handled:

1. -n : prints out the number of each line in the file.

2. -e : prints a ‘$’ symbol at the end of each line in the file.

Edge Cases:

1. cat file_doesnt_exist
   Prints out “File not found” along with the filename that is not found.

2. cat directory1/file1 directory2/file2
   Prints out both files file1 and file2 provided the paths exist. If a path does not exist then, then it is printed out that the path/file does not exist, however the path which does exist will work as normal.

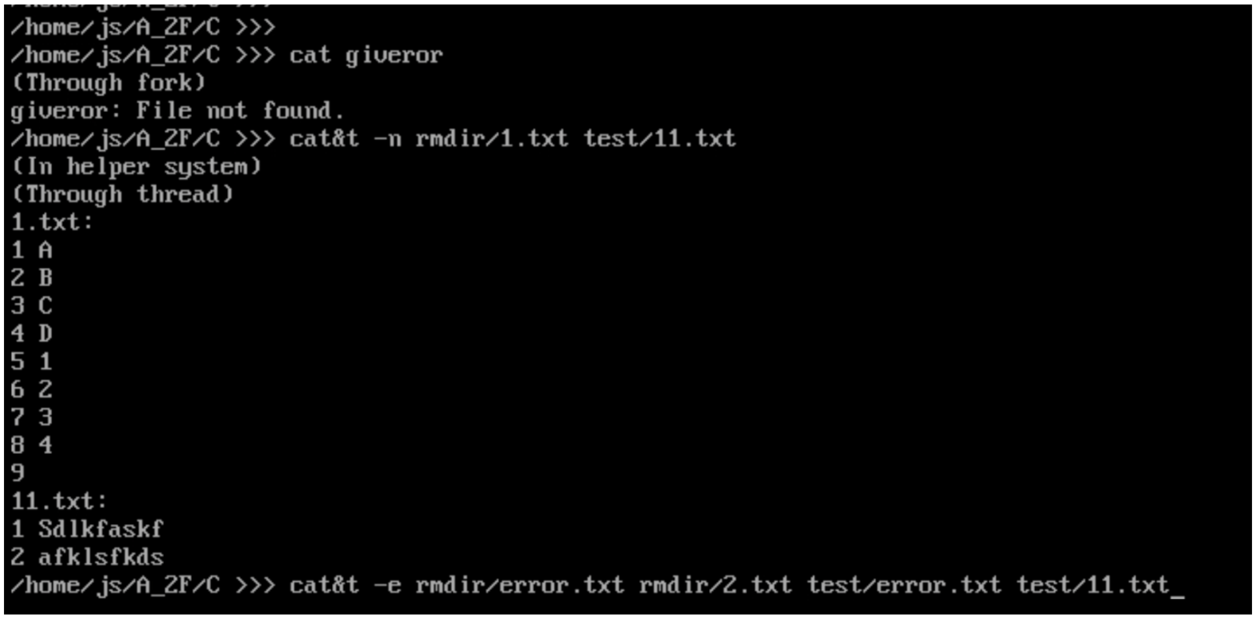
3. cat -invalid_flag
   Prints out that the flag is invalid along with the flag.

4. cat directory/ filename
   Since spaces are treated as arguments (to provide protection and accuracy), cat will not be able to read this file as ‘ ‘ will be treated as an invalid path name.

5. cat
   If no argument is provided then it prints out the same.

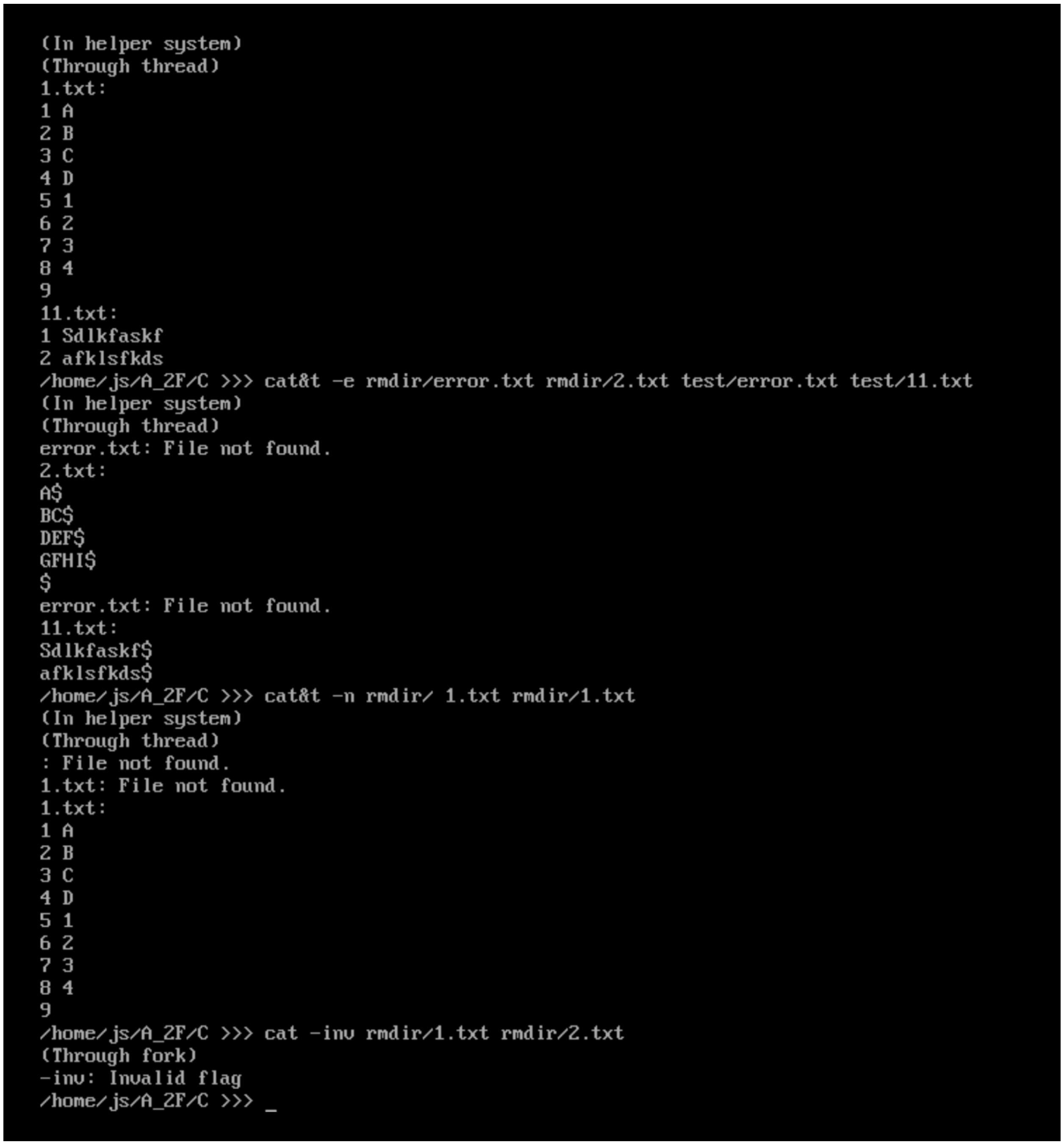
6. cat directory/
   Gracefully ignores this as it was expecting a filename but never got one.

Basic Linux/Unix Shell



```
/home/js/A_2F/C >>>
/home/js/A_2F/C >>> cat giveror
(Through fork)
giveror: File not found.
/home/js/A_2F/C >>> cat&t -n rmdir/1.txt test/11.txt
(In helper system)
(Through thread)
1.txt:
1 A
2 B
3 C
4 D
5 1
6 2
7 3
8 4
9
11.txt:
1 Sdlkfaskf
2 afklsfkds
/home/js/A_2F/C >>> cat&t -e rmdir/error.txt rmdir/2.txt test/error.txt test/11.txt_
```

Basic Linux/Unix Shell



```
(In helper system)
(Through thread)
1.txt:
1 A
2 B
3 C
4 D
5 1
6 2
7 3
8 4
9
11.txt:
1 Sdlkfaskf
2 afklsfkds
/home/js/A_2F/C >>> cat&t -e rmdir/error.txt rmdir/2.txt test/error.txt test/11.txt
(In helper system)
(Through thread)
error.txt: File not found.
2.txt:
A$
BC$
DEF$
GFHI$
$
error.txt: File not found.
11.txt:
Sdlkfaskf$
afklsfkds$
/home/js/A_2F/C >>> cat&t -n rmdir/ 1.txt rmdir/1.txt
(In helper system)
(Through thread)
: File not found.
1.txt: File not found.
1.txt:
1 A
2 B
3 C
4 D
5 1
6 2
7 3
8 4
9
/home/js/A_2F/C >>> cat -inv rmdir/1.txt rmdir/2.txt
(Through fork)
-inv: Invalid flag
/home/js/A_2F/C >>> _
```

Basic Linux/Unix Shell

7. mkdir

Flags handled:

1. -v : prints out a message informing the user of whether the directory has been created or not.

2. -p : allows creation of parent directories as per requirement.

Edge Cases:

1. mkdir -p directory_exists/directory_doesnt_exist
   This will not throw any errors (that is, if a directory already exists).

2. mkdir directory_already_exists
   Prints out that the directory was not able to be created (since it already exists).

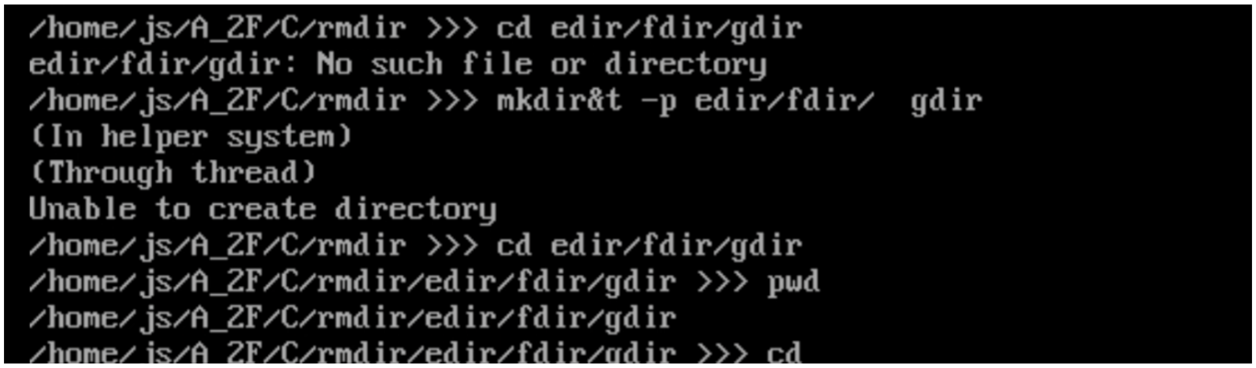
3. mkdir -invalid_flag
   Prints out that the flag is invalid along with the flag.

4. mkdir -p dir1/dir2/ dir3
   Creates the directories properly, however it tells the user that it could not create a directory due to the blank spaces.

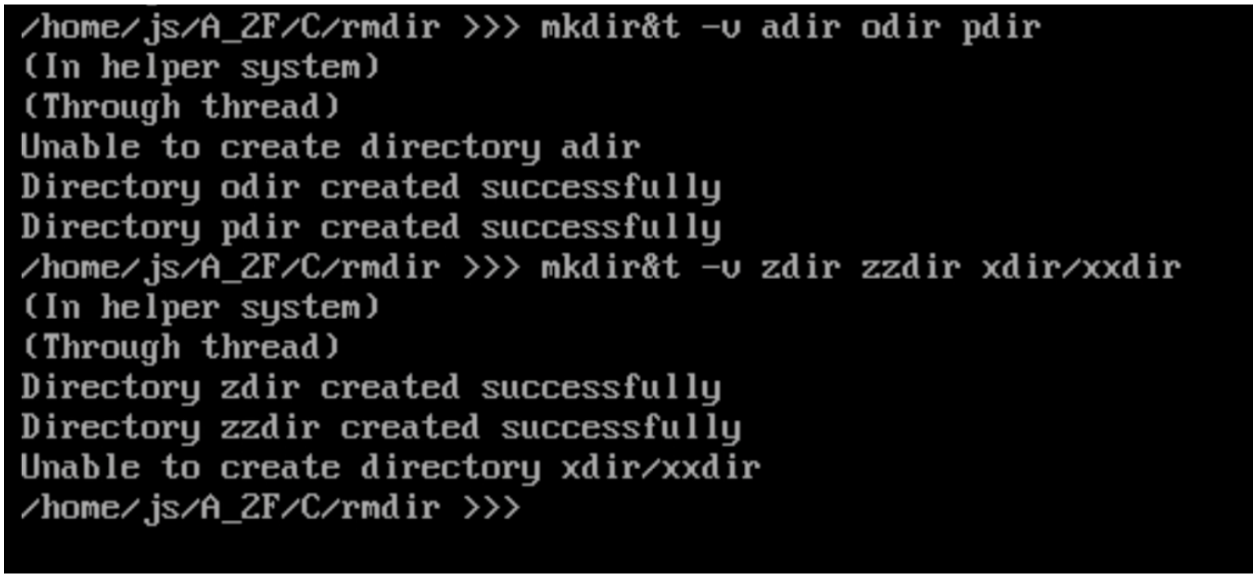
5. mkdir&t -v adir bdir cdir/ddir
   Prevents the last directory from being created but creates the other two directories.

6. mkdir
   If no argument is provided then it prints out the same.

Basic Linux/Unix Shell



```
/home/js/A_ZF/C/rmdir >>> cd edir/fdir/gdir
edir/fdir/gdir: No such file or directory
/home/js/A_ZF/C/rmdir >>> mkdir&t -p edir/fdir/  gdir
(In helper system)
(Through thread)
Unable to create directory
/home/js/A_ZF/C/rmdir >>> cd edir/fdir/gdir
/home/js/A_ZF/C/rmdir/edir/fdir/gdir >>> pwd
/home/js/A_ZF/C/rmdir/edir/fdir/gdir
/home/js/A_ZF/C/rmdir/edir/fdir/gdir >>> cd
```



```
/home/js/A_ZF/C/rmdir >>> mkdir&t -v adir odir pdir
(In helper system)
(Through thread)
Unable to create directory adir
Directory odir created successfully
Directory pdir created successfully
/home/js/A_ZF/C/rmdir >>> mkdir&t -v zdir zzdir xdir/xxdir
(In helper system)
(Through thread)
Directory zdir created successfully
Directory zzdir created successfully
Unable to create directory xdir/xxdir
/home/js/A_ZF/C/rmdir >>>
```

Basic Linux/Unix Shell

8. rm

Flags handled:

1. -i : Asks the user for confirmation before deleting a file, if ‘y’ is entered then the file is deleted otherwise it is not.

2. -v : prints out a message informing the user of whether the file has been deleted or not

Edge Cases:

1. rm file_doesnt_exist
   Prints out that the file does not exist and therefore could not be deleted

2. rm directory1/file1 directory2/file2
   Deletes both file1 and file2 from their respective directories provided the paths exist.
   If a path does not exist, then it is printed out that the path does not exist, however the path which does exist will work as normal.

3. rm -invalid_flag
   Prints out that the flag is invalid along with the flag.

4. rm directory_name
   Doesn’t allow deletion of directories.

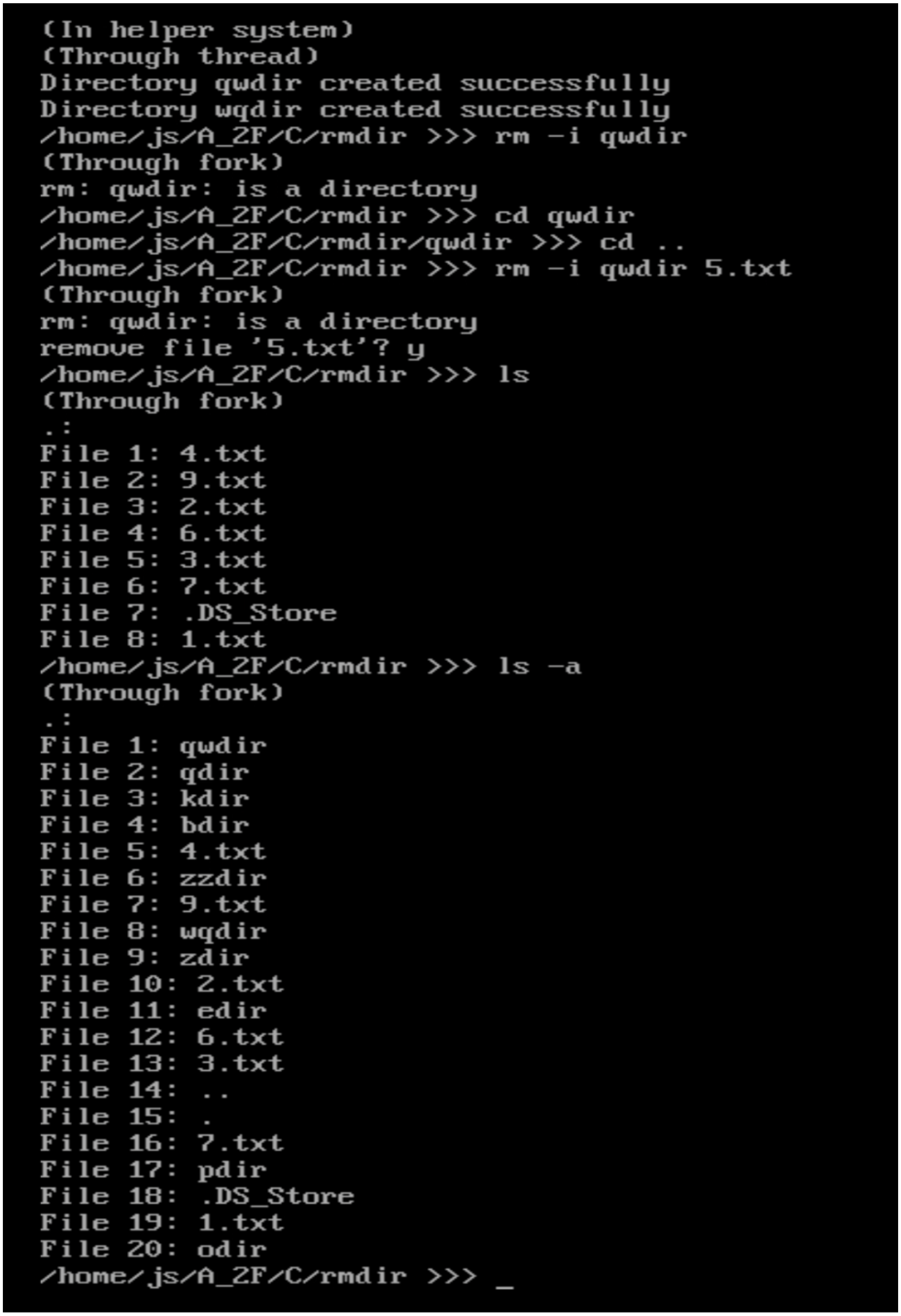
5. rm directory/ filename
   Since spaces are treated as arguments (to provide protection), rm will not be able to delete this file.

Basic Linux/Unix Shell

6. rm

   If no argument is provided then it prints out the same.

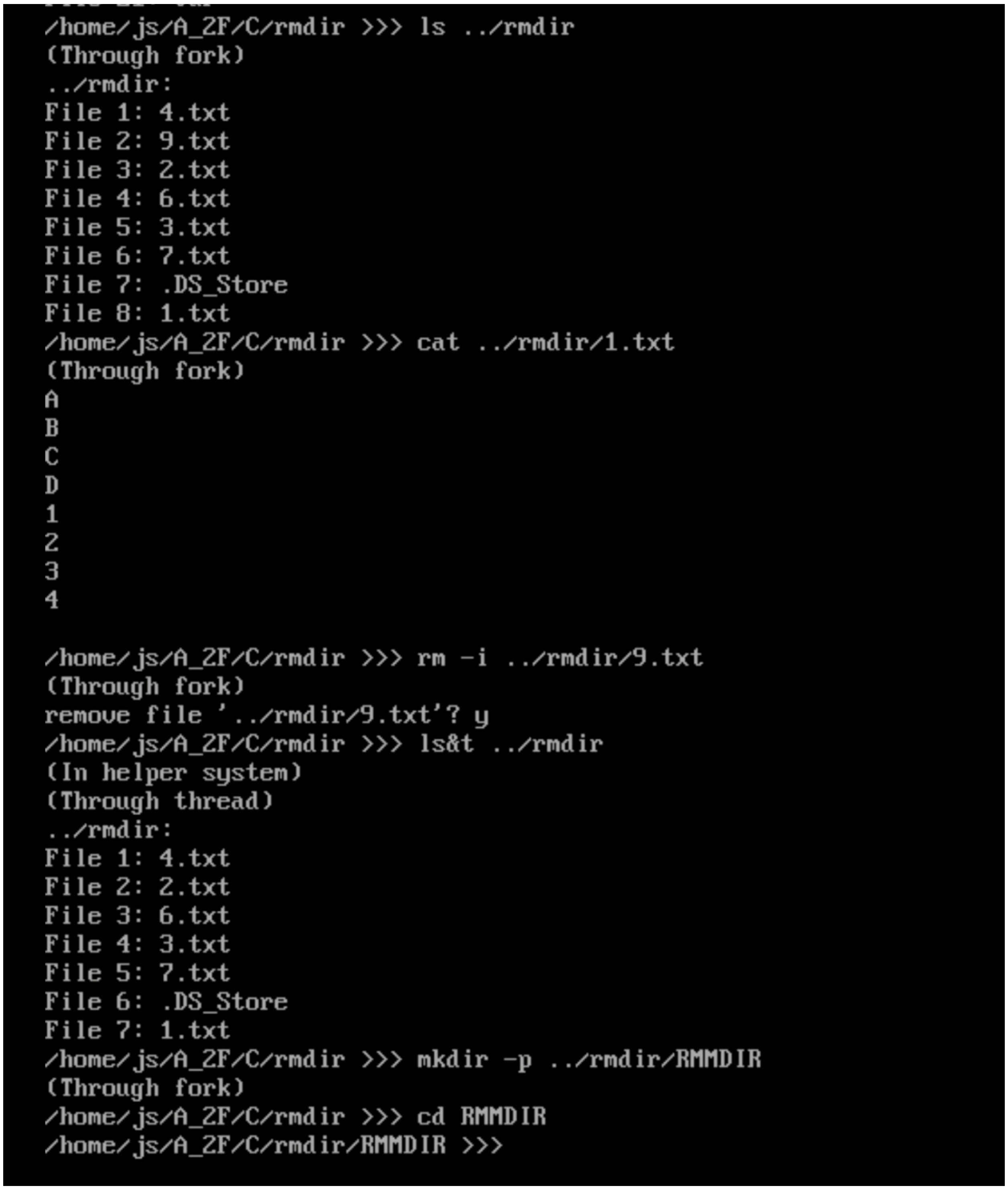
Basic Linux/Unix Shell



```
(In helper system)
(Through thread)
Directory qwdir created successfully
Directory wqdir created successfully
/home/js/A_2F/C/rmdir >>> rm -i qwdir
(Through fork)
rm: qwdir: is a directory
/home/js/A_2F/C/rmdir >>> cd qwdir
/home/js/A_2F/C/rmdir/qwdir >>> cd ..
/home/js/A_2F/C/rmdir >>> rm -i qwdir 5.txt
(Through fork)
rm: qwdir: is a directory
remove file '5.txt'? y
/home/js/A_2F/C/rmdir >>> ls
(Through fork)
.:
File 1: 4.txt
File 2: 9.txt
File 3: 2.txt
File 4: 6.txt
File 5: 3.txt
File 6: 7.txt
File 7: .DS_Store
File 8: 1.txt
/home/js/A_2F/C/rmdir >>> ls -a
(Through fork)
.:
File 1: qwdir
File 2: qdir
File 3: kdir
File 4: bdir
File 5: 4.txt
File 6: zzdir
File 7: 9.txt
File 8: wqdir
File 9: zdir
File 10: 2.txt
File 11: edir
File 12: 6.txt
File 13: 3.txt
File 14: ..
File 15: .
File 16: 7.txt
File 17: pdir
File 18: .DS_Store
File 19: 1.txt
File 20: odir
/home/js/A_2F/C/rmdir >>> _
```

Basic Linux/Unix Shell

General Test Case:



```
/home/js/A_2F/C/rmdir >>> ls ../rmdir
(Through fork)
../rmdir:
File 1: 4.txt
File 2: 9.txt
File 3: 2.txt
File 4: 6.txt
File 5: 3.txt
File 6: 7.txt
File 7: .DS_Store
File 8: 1.txt
/home/js/A_2F/C/rmdir >>> cat ../rmdir/1.txt
(Through fork)
A
B
C
D
1
2
3
4

/home/js/A_2F/C/rmdir >>> rm -i ../rmdir/9.txt
(Through fork)
remove file '../rmdir/9.txt'? y
/home/js/A_2F/C/rmdir >>> ls&t ../rmdir
(In helper system)
(Through thread)
../rmdir:
File 1: 4.txt
File 2: 2.txt
File 3: 6.txt
File 4: 3.txt
File 5: 7.txt
File 6: .DS_Store
File 7: 1.txt
/home/js/A_2F/C/rmdir >>> mkdir -p ../rmdir/RMMDIR
(Through fork)
/home/js/A_2F/C/rmdir >>> cd RMMDIR
/home/js/A_2F/C/rmdir/RMMDIR >>>
```