BHARATI VIDYEETH'S

INSTITUTE OF COMPUTER PLICATIONS & MANAGEMENT

(Affiliated to Guru Gobind Singh Indraprastha University, Approved by AICTE, New Delhi)

Operating System With Linux Lab

(MCA-167)

Practical File

Submitted To:

Dr. Sunil Pratap Singh

Submitted By:

Aryan Chandra

(06711604423)

INDEX

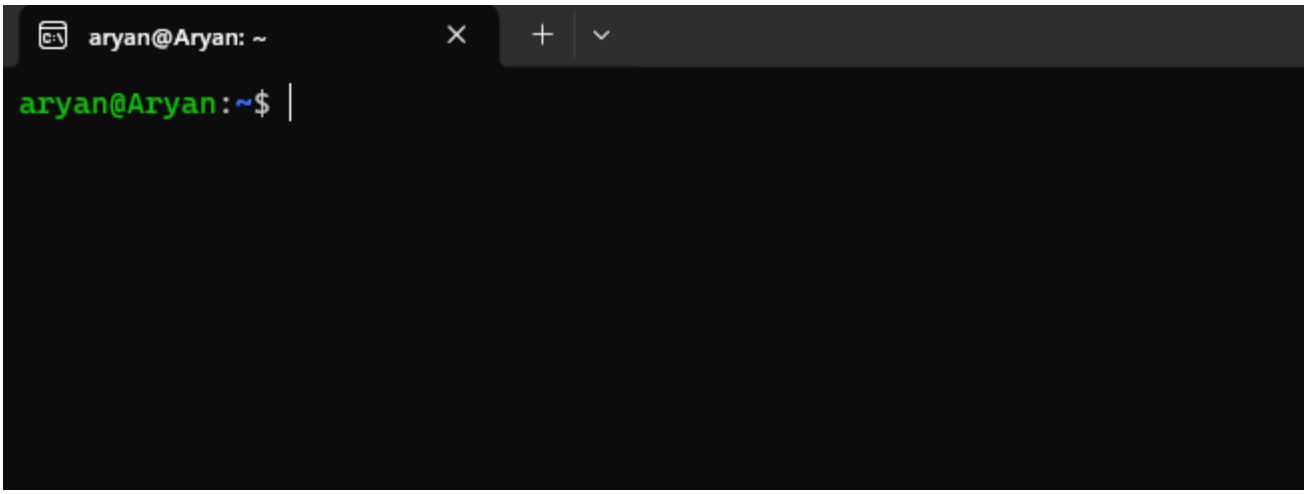
S. No.	Problem Description	Date of Execution	Sign.
2	Run ps and note the PID of your shell. Log out and login again and run ps again. What do you observe?		
3	Enter the following commands, and note yourobservations: (i) who and tty, (ii) tput clear, (iii) id, (iv) ps and echo $$.		
4	Run the following commands, and then invoke ls.What do you conclude? echo > README [Enter] echo > readme [Enter]		
5	Create a directory, and change to that directory. Next,create another directory in the new directory, and then change to that directory too. Now, run $ cd without any arguments followed by pwd. What do you conclude?		
6	Create a file mca containing the words “Hello MCA Class!”. Now create a directory bvicam, and then runmv mca bvicam. What do you observe when you run both ls and ls bar?		
7	Run $ who am i and then interpret the output.		
8	Find out whether the following commands are internal or external: echo, date, pwd, and ls		
9	Display the current date in the form dd/mm/yyyy.		
10	Both of the following commands try to open the filemca, but the error messages are a little different. What could be the reason? $ cat mca cat: mca: No such file or directory $ cat < mca bash: mca: No such file or directory		

11	Run the following commands, and discuss theiroutput? (a) $ uname (b) $ passwd (c) $ echo $SHELL (d) $ man man (e) $ which echo (f) $ type echo (g) $ whereis ls (h) $ cd (i) $ cd $HOME (j) $ cd ~		
12	Frame ls command to (i) mark directories and executables separately, and (ii) also display hidden files.		
13	Find out the result of following: $ cat mca mca mca		
14	Run the following and determine which commands willwork? Explain with reasons. (a) $ mkdir a/b/ (b) $ mkdir a a/b (c) $ rmdir a/b/c (d) $ rmdir a a/b (e) $ mkdir /bin/mca		
15	How does the command mv mca1 mca2 behave, where both mca1 and mca2 are directories, when (i) mca2 exists, (ii) (ii) mca2 doesn‟t exist?		
16	Assuming that you are positioned in the directory /home/bvicam, what are these commands presumedto do, and explain whether they will work at all: (a) $ cd ../..		

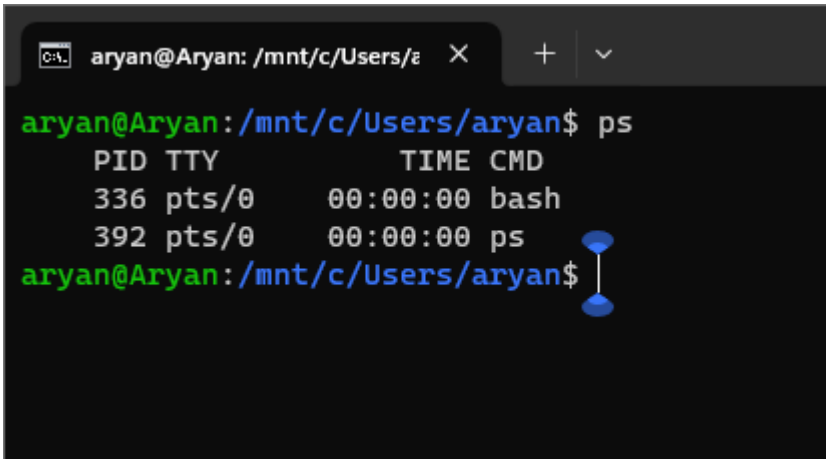
	(b) $ mkdir ../bin (c) $ rmdir . . (d) $ ls ..		
17	ply Peterson algorithm for solving the critical section problem with C/Java multi-threaded programming. Assumepropriate code snippet for critical section		
18	ply Bakery algorithm for synchronization of processes/threadsin a C/Java program. Assume propriate code snippet for critical section.		
19	Write C/Java program to simulate and solve theProducer- Consumer problem		
20	Implement Semhore(s) in a C/Java-multithreaded program to simulate the working and solution of Reader-Writer problem. Assume multiple readers and writers		
21	Create a zombie process and an orphan process in a „C‟ program with propriate system calls.		
22	Write a „C‟ program which creates a new process andallows both, child and parent, to report their identification numbers (ids). The parent process should wait for the termination of the child process		
23	Write two „C‟ programs (A.c and B.c) where one program (A.c) creates a child process and then that child process executes the code of other program (B.c). The logic of program „B.c‟ is to generate all the prime numbers within the specified limit.		
24	Write an propriate „C‟ program which implementsthe conceptof dynamic memory allocation (use of malloc(), calloc(), realloc(), and free() system call)		
25	Create a text file, named as „courses.txt‟ that containsthe following four lines: Java Programming Operating System Discrete Structure Write a „C‟ program that		

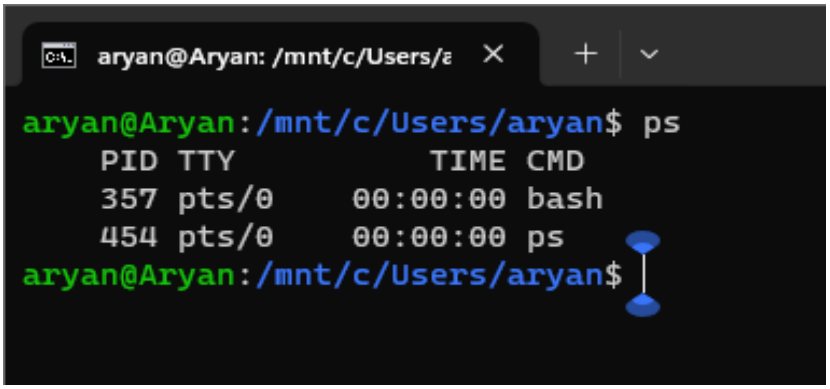
	forks three other processes. After forking, the parent process goes into wait state and waits for the childrento finish their execution. Each child process reads a line from the „course.txt“ file (Child 1 Reads Line 1, Child 2 Reads Line 2, and Child 3 Reads Line 3) and each prints the respective line. The lines can be printed in any order.		
26	Write a „C“ program (using propriate system calls ofLinux) that generates „n“ integers and stores them in atext file, named as „All.txt“. Then, retrieve the stored integers from this file and copy to “Odd.txt” and „Even.txt“ based upon the type of number, i.e. if theretrieved integer if odd number then store in „Odd.txt“ file or if the retrieved integer is even then store in „Even.txt“ file. Finally, display the contents of all three files on the screen		
27	Write a program in „C“ which accepts the file or directory name and permission (access rights) fromthe user and then changes the access rights accordingly. Use propriate system call(s) of Linux.		
28	Write a „C“ program (using propriate system calls ofLinux)which generates and stores the characters from „a“ to „z“. Then, display the stored characters in alternative manner, like: a, c, e, g, …, etc.		
29	Write a „C“ program (using propriate system calls ofLinux) which receives roll number and names of „n“ students, from theuser one-by-one and then stores them in a text file, named as „Student.txt“. After inserting all „n“ roll numbers and names, display the contents of file. Also, display the access rights of the file „Student.txt“		
30	Demonstrate the use of following system calls bywriting an propriate „C“ program. (a) lseek() (b) chmod() (c) umask() (d) access() (e) utime()		

Program 1



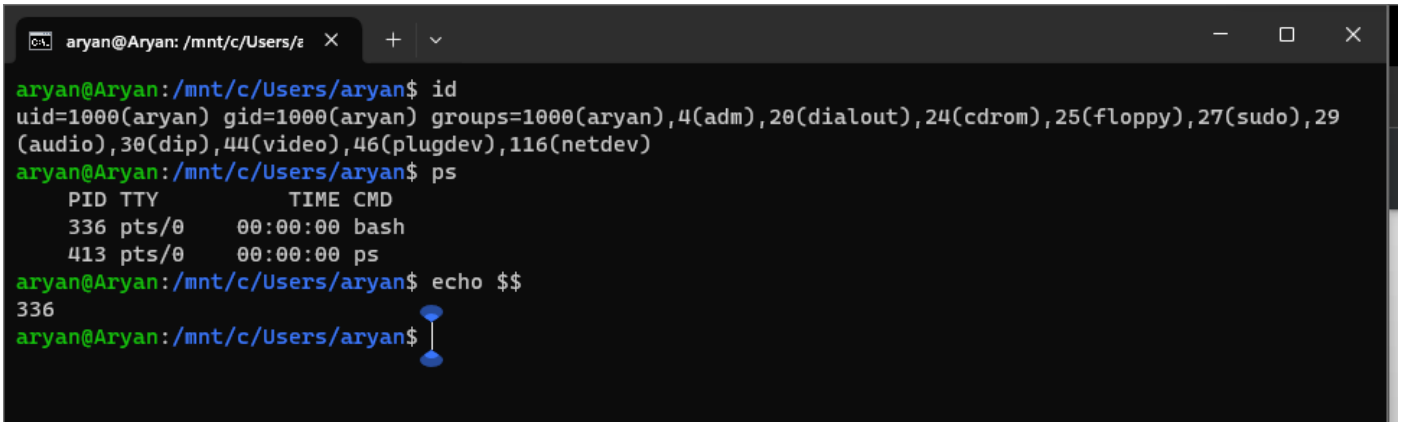
Program 2

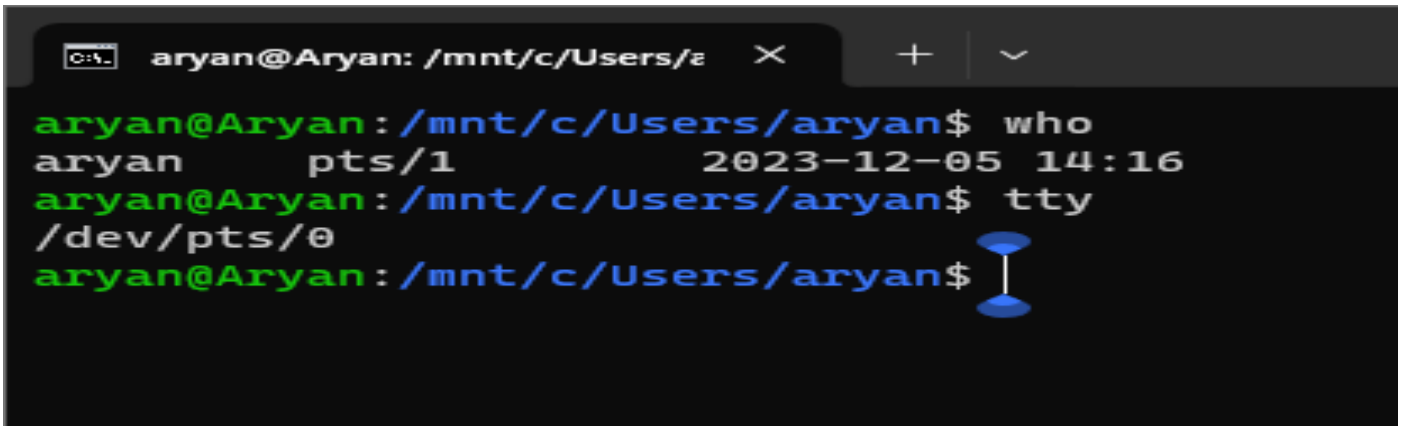




Yes , New Program ID is assigned every time Log in Log Out process is performed

Program 3





The WHO command outputs a list of users currently logged into the system.

The TTY command displays the name of the current terminal device. The terminal device is the physical or virtual interface through which a user interacts with the computer.

The TTY command clears the screen, removing all text from the terminal window.

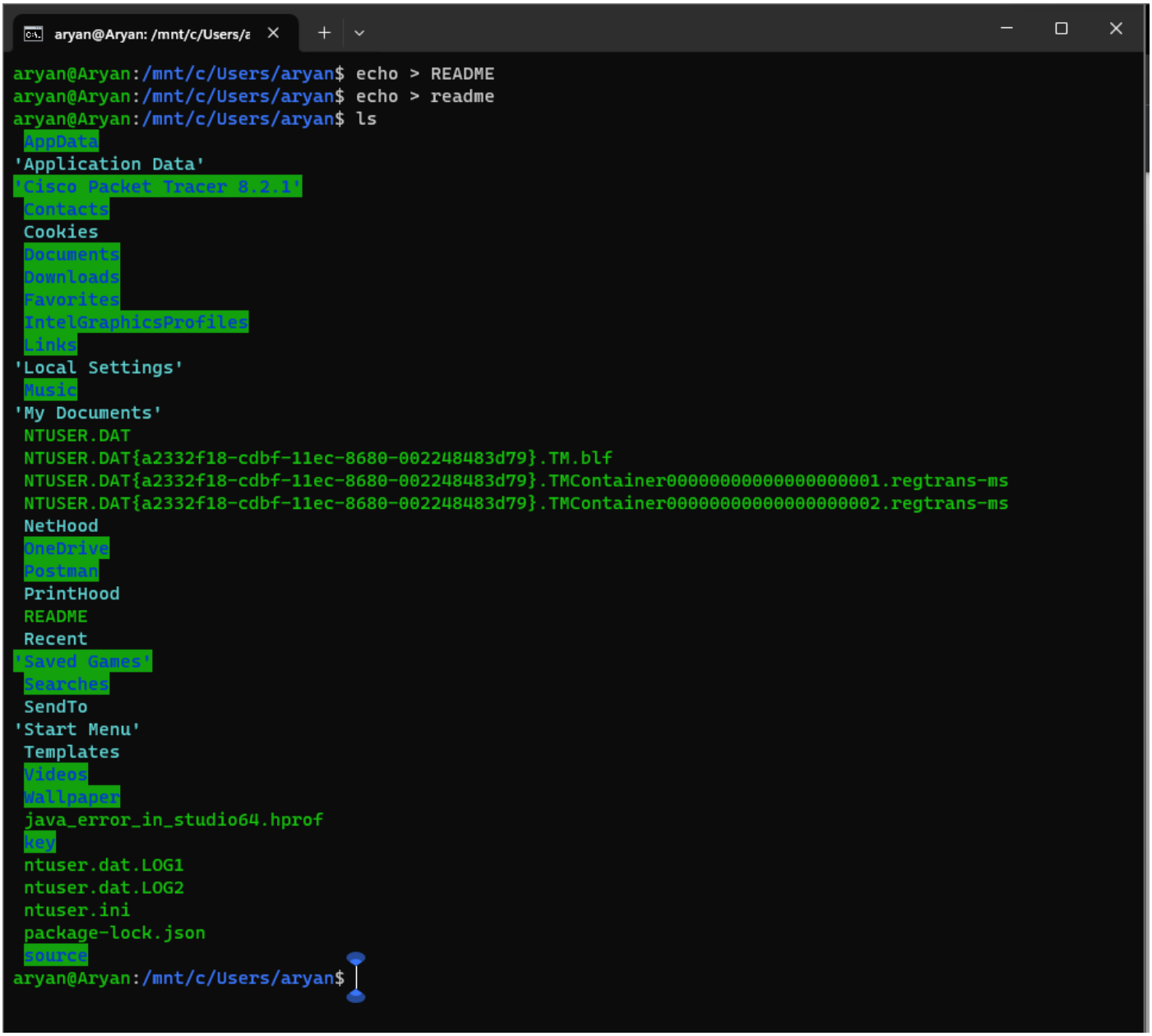
The ID command displays information about the current user and group. The output includes the following:

- User ID (UID)
- Group ID (GID)
- Effective user ID (EUID)
- Effective group ID (EGID)
- Groups to which the user belongs
- User name

- Group name

The PS command lists all processes currently running on the system. The ECHO $$ command outputs the process ID (PID) of the current shell. When the PS command is executed, it also includes the PID of the echo command.

Program 4



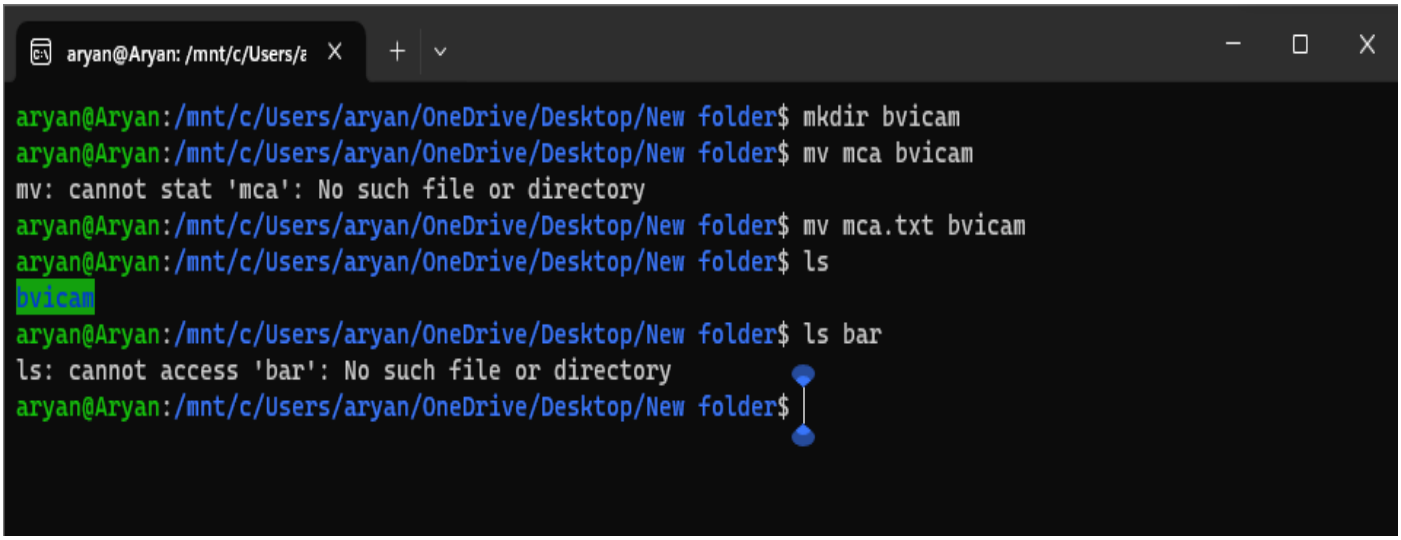
Running the commands echo > README and echo > readme followed by ls will result in two files named README and readme being created in the current directory. This is because the > operator redirects the output of the echo command to the specified file. Since the echo command does not produce any output when invoked without arguments, empty files are created.

Program 5

```
aryan@Aryan:/mnt/c/Users/aryan$ mkdir newDirectory
aryan@Aryan:/mnt/c/Users/aryan$ cd newDirectory/
aryan@Aryan:/mnt/c/Users/aryan/newDirectory$ mkdir newDirectory2
aryan@Aryan:/mnt/c/Users/aryan/newDirectory$ cd newDirectory2/
aryan@Aryan:/mnt/c/Users/aryan/newDirectory/newDirectory2$ cd
aryan@Aryan:~$ pwd
/home/aryan
aryan@Aryan:~$
```

CD command without argument returns working directory to home directory and when we run PWD command the location of currently working directory is printed.

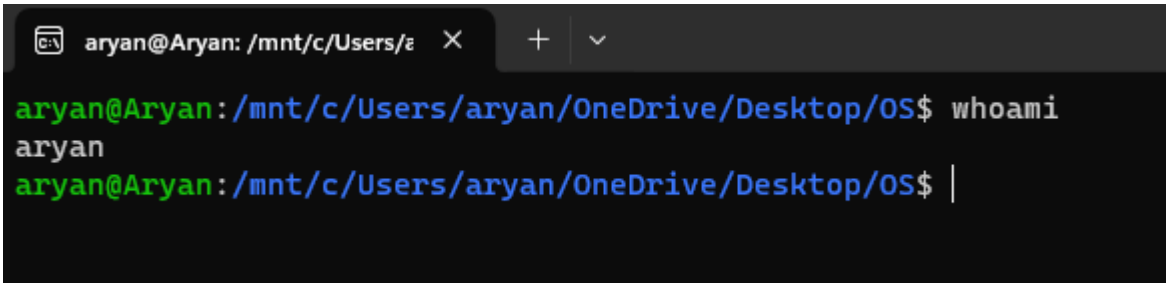
Program 6



When we run LS command the content of working directory is printed .

LS Bar will return an error as bar is not any directory or file.

Program 7



Whoami command prints the name of currently logged in user.

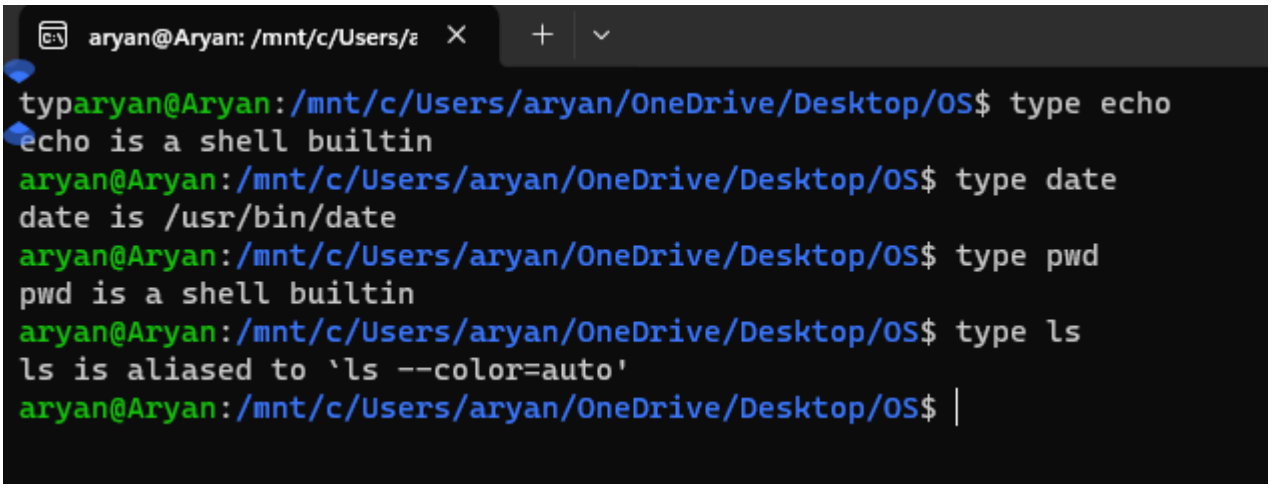
Program 8

ECHO is an internal command.

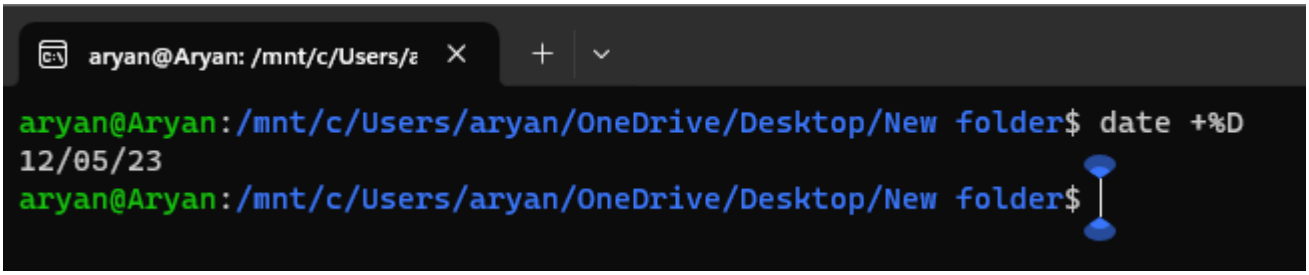
DATE is an external command.

PWD is an internal command.

LS is an external command.



Program 9


aryan@Aryan: /mnt/c/Users/a
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/New folder$ date +%D
12/05/23
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/New folder$

Program 10

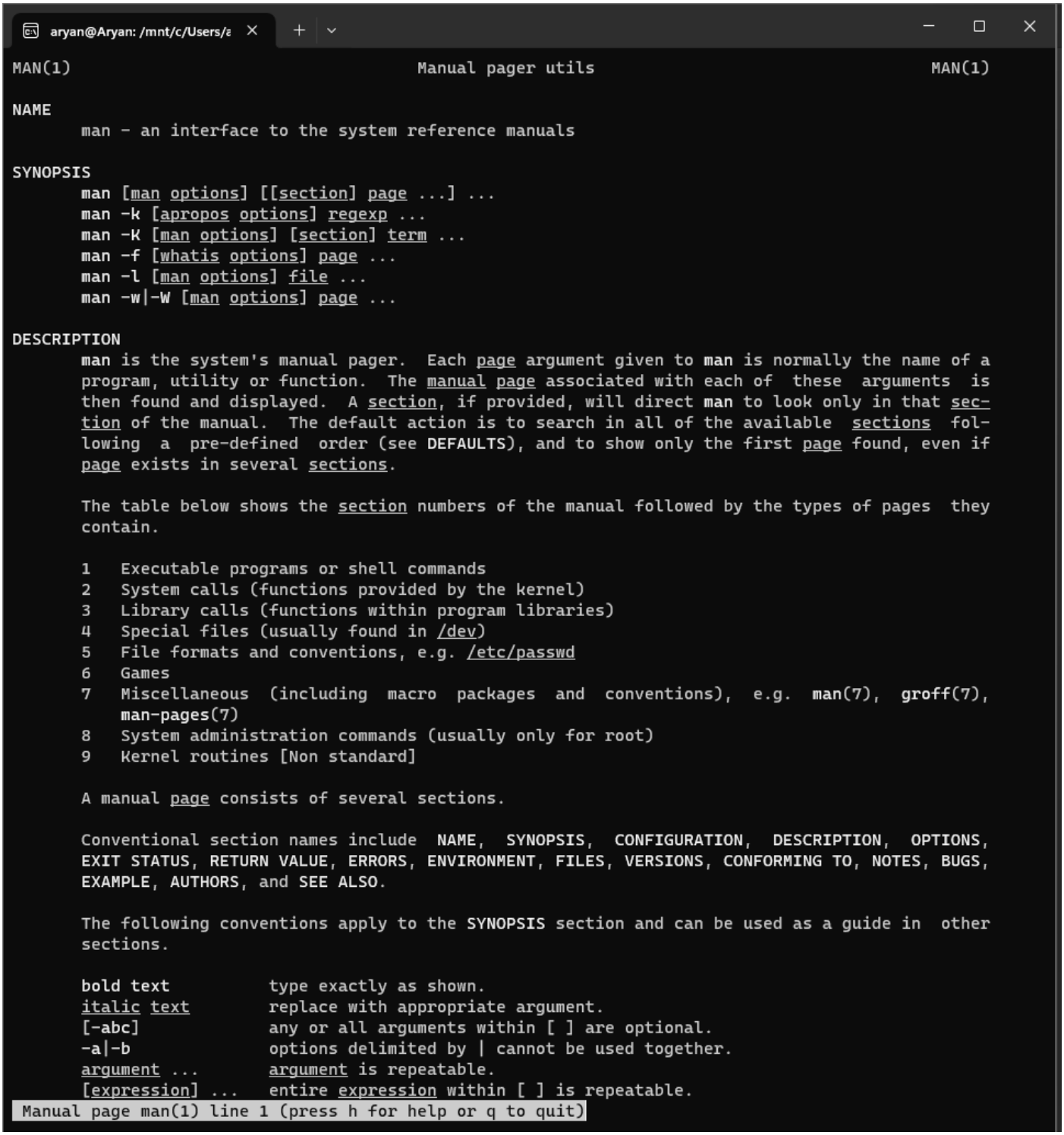
cat mca

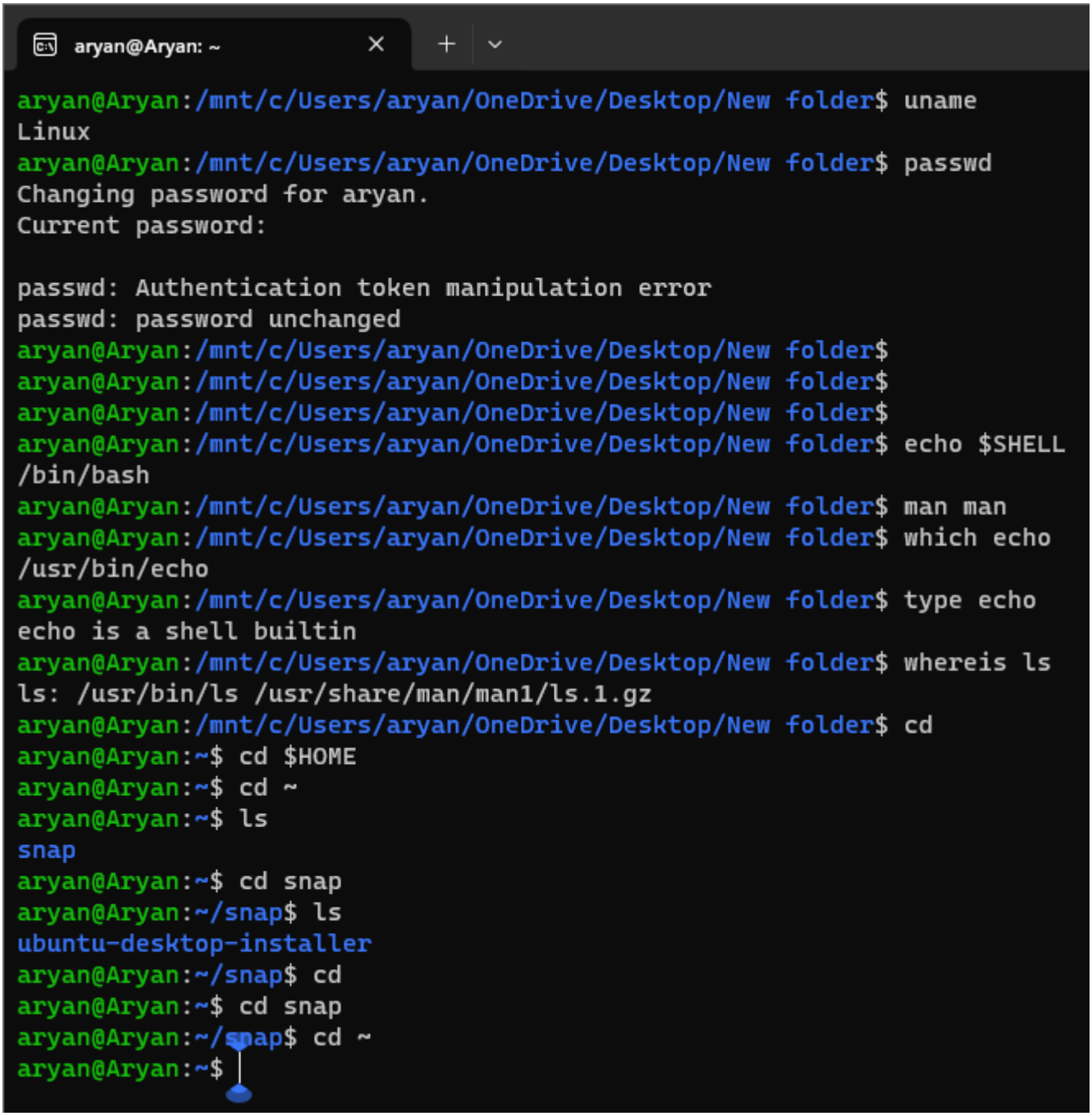
The cat command is used to display the contents of a file. When you run the command cat mca, the shell attempts to open the file mca in the current directory. If the file does not exist, the shell will display an error message stating that the file was not found. The specific error message "No such file or directory" indicates that the file does not exist at the specified location (the current directory).

cat < mca

The redirection operator > is used to redirect the output of a command to a file. In this case, the command cat is attempting to read the contents of the file mca and redirect the output to the standard output (the terminal window). However, if the file mca does not exist, the cat command will have no input to read, and the shell will display an error message stating that the file was not found. The specific error message "cat: mca: No such file or directory" indicates that the file mca does not exist and that the cat command was unable to redirect its output to the file.

Program 11


aryan@Aryan: /mnt/c/Users/a
MAN(1) Manual pager utils MAN(1)
NAME
man - an interface to the system reference manuals
SYNOPSIS
man [man options] [[section] page ...] ...
man -k [apropos options] regexp ...
man -K [man options] [section] term ...
man -f [whatis options] page ...
man -l [man options] file ...
man -w|-W [man options] page ...
DESCRIPTION
man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order (see DEFAULTS), and to show only the first page found, even if page exists in several sections.
The table below shows the section numbers of the manual followed by the types of pages they contain.
1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions, e.g. /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7), man-pages(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]
A manual page consists of several sections.
Conventional section names include NAME, SYNOPSIS, CONFIGURATION, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUE, ERRORS, ENVIRONMENT, FILES, VERSIONS, CONFORMING TO, NOTES, BUGS, EXAMPLE, AUTHORS, and SEE ALSO.
The following conventions apply to the SYNOPSIS section and can be used as a guide in other sections.
bold text type exactly as shown.
italic text replace with appropriate argument.
[-abc] any or all arguments within [ ] are optional.
-a|-b options delimited by | cannot be used together.
argument ... argument is repeatable.
[expression] ... entire expression within [ ] is repeatable.
Manual page man(1) line 1 (press h for help or q to quit)



UNAME command display the system’s kernel and operating system.

PASSWD command is used to change the password.

Echo $SHELL displays the location of SHELL.

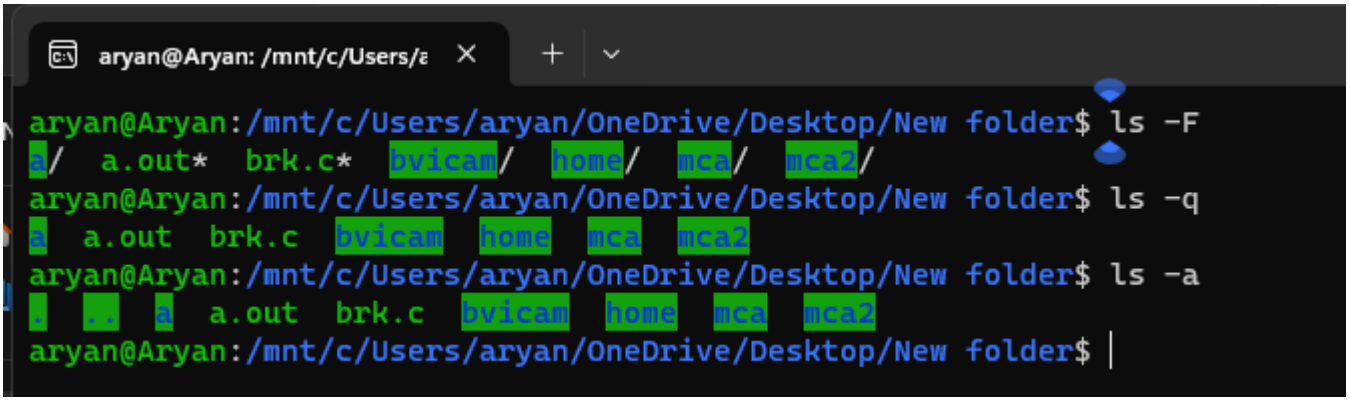
MAN MAN command shows the user guide form MAN command.

TYPE ECHO command shows weather this is internal command or external command.

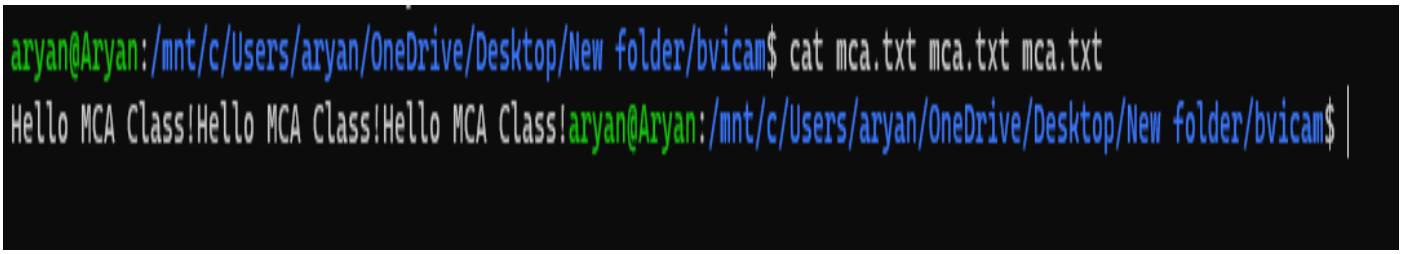
WHEREIS LS shows the location of LS executable.

CD $HOME , CD , CD~ both perform same task i.e., return to home working directory.

Program 12


aryan@Aryan: /mnt/c/Users/a
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/New folder$ ls -F
a/ a.out* brk.c* bvicam/ home/ mca/ mca2/
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/New folder$ ls -q
a a.out brk.c bvicam home mca mca2
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/New folder$ ls -a
. .. a a.out brk.c bvicam home mca mca2
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/New folder$

Program 13


aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/New folder/bvicam$ cat mca.txt mca.txt mca.txt
Hello MCA Class!Hello MCA Class!Hello MCA Class!aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/New folder/bvicam$

Program 14

(a) $ mkdir a/b/

This command will not work because it tries to create a subdirectory b inside a non-existent directory a. The mkdir command requires that the parent directory already exists, unless the -p option is used. To create both a and b directories, you can use the command: $ mkdir -p a/b/1

(b) $ mkdir a a/b

This command will partially work because it will create the directory a, but it will fail to create the subdirectory b inside a. The reason is the same as in (a), the mkdir command does not create intermediate directories by default. To create both a and b directories, you can use the command: $ mkdir -p a a/b1

© $ rmdir a/b/c

This command will work only if the directory a/b/c exists and is empty. The rmdir command removes empty directories only. If the directory a/b/c is not empty or does not exist, the command will fail and display an error message. To remove a directory with its contents, you can use the command: $ rm -r a/b/c2

(d) $ rmdir a a/b

This command will not work because it tries to remove the directory a before removing its subdirectory b. The rmdir command does not remove non-empty directories, so it will fail and display an error message. To remove both a and b directories, you can use the command: $ rmdir a/b a2

(e) $ mkdir /bin/mca

This command will not work because it tries to create a directory under /bin, which is a system directory that requires root privileges to modify. The mkdir command will fail and display a permission denied error message. To create a directory under /bin, you need to use the sudo command and provide the root password: $ sudo mkdir /bin/mca3

Program 15

(i) If mca2 exists and is a directory, then the command mv mca1 mca2 will move the directory mca1 inside the directory mca2. This means that mca1 will no longer exist in its original location, but will become a subdirectory of mca2. The contents of mca1 will remain unchanged. For example, if you have the following directories:

(ii) If mca2 does not exist, then the command mv mca1 mca2 will rename the directory mca1 to mca2. This means that mca1 will no longer exist, but will be replaced by mca2. The contents of mca1 will remain unchanged.

Program 16

(a) $ cd …/…

This command will change the current directory to the parent directory of the parent directory of /home/bvicam, which is /.

(b) $ mkdir …/bin

This command will create a new directory named bin in the parent directory of /home/bvicam, which is /home.

© $ rmdir …

This command will try to remove the parent directory of /home/bvicam, which is /home. This command will not work because rmdir can only remove empty directories, and /home is likely to contain other files and directories besides bvicam.

(d) $ ls .

This command will list the files and directories in the current directory, which is /home/bvicam.

Program 17

```
#include <stdio.h>
#include <pthread.h>

int flag[2] = {0};
int turn = 0;

void enterCriticalSection(int threadId) {
  flag[threadId] = 1;

  while (flag[1 - threadId] == 1 && turn == 1 - threadId) {}
}

void exitCriticalSection(int threadId) {
  flag[threadId] = 0;
  turn = 1 - threadId;
}

void* criticalSectionFunction(void* arg) {
  int threadId = *((int*)arg);

  enterCriticalSection(threadId);

  printf("Thread %d is in the critical section.\n", threadId);

  exitCriticalSection(threadId);

  return NULL;
}
```

```
int main() {
  pthread_t thread1, thread2;

  pthread_create(&thread1, NULL, criticalSectionFunction, (void*)&0);
  pthread_create(&thread2, NULL, criticalSectionFunction, (void*)&1);

  pthread_join(thread1, NULL);
  pthread_join(thread2, NULL);

  return 0;
}
```

Program 18

```
#include <stdio.h>
#include <pthread.h>

int choosing[2] = {0};
int ticket[2] = {0};

void enterCriticalSection(int processId) {
  choosing[processId] = 1;

  int max = 0;
  for (int i = 0; i < 2; i++) {
    if (ticket[i] > max) {
      max = ticket[i];
    }
  }

  ticket[processId] = max + 1;

  choosing[processId] = 0;

  for (int i = 0; i < 2; i++) {
    if (i != processId) {
      while (choosing[i] == 1) {}

      while (ticket[i] <= ticket[processId] && (i < processId || (i == processId && ticket[i] == ticket[processId]))) {}
    }
  }
}
```

```
void exitCriticalSection(int processId) {
  ticket[processId] = 0;
}

void* criticalSectionFunction(void* arg) {
  int processId = *((int*)arg);
enterCriticalSection(processId);
printf("Process %d is in the critical section.\n", processId);
exitCriticalSection(processId);
 return NULL;
}

int main() {
  pthread_t thread1, thread2;
  pthread_create(&thread1, NULL, criticalSectionFunction, (void*)&0);
  pthread_create(&thread2, NULL, criticalSectionFunction, (void*)&1);
  pthread_join(thread1, NULL);
  pthread_join(thread2, NULL);

  return 0;
}
```

Program 19

```
Producer 1 produced item 83 at index 0
Producer 1 produced item 77 at index 1
Producer 2 produced item 86 at index 2
Producer 2 produced item 15 at index 3
Producer 2 produced item 93 at index 4
Producer 2 produced item 35 at index 5
Producer 2 produced item 86 at index 6
Producer 2 produced item 92 at index 7
Producer 2 produced item 49 at index 8
Producer 2 produced item 21 at index 9
Consumer 2 consumed item 83 at index 0
Consumer 1 consumed item 77 at index 1
Consumer 2 consumed item 86 at index 2
Consumer 2 consumed item 15 at index 3
Consumer 2 consumed item 93 at index 4
Consumer 2 consumed item 35 at index 5
Consumer 2 consumed item 86 at index 6
Consumer 2 consumed item 92 at index 7
Consumer 2 consumed item 49 at index 8
Producer 1 produced item 62 at index 0
Producer 1 produced item 90 at index 1
Producer 1 produced item 59 at index 2
Producer 1 produced item 63 at index 3
Producer 1 produced item 26 at index 4
Producer 1 produced item 40 at index 5
Producer 1 produced item 26 at index 6
Producer 1 produced item 72 at index 7
Producer 2 produced item 27 at index 8
Consumer 1 consumed item 21 at index 9
Consumer 2 consumed item 62 at index 0
```

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semhore.h>

#define BUFFER_SIZE 10

struct buffer {
    int data[BUFFER_SIZE];
    int in;
```

```
    int out;
};

struct buffer buffer;

sem_t mutex;
sem_t empty;
sem_t full;

void *producer(void *arg) {
    int item;
    while (1) {
        item = rand() % 100;
        sem_wait(&empty);
        sem_wait(&mutex);
        buffer.data[buffer.in] = item;
        printf("Producer %d produced item %d at index %d\n", (int)arg, item, buffer.in);
        buffer.in = (buffer.in + 1) % BUFFER_SIZE;
        sem_post(&mutex);
        sem_post(&full);
    }
}

void *consumer(void *arg) {
    int item;
    while (1) {
        sem_wait(&full);
        sem_wait(&mutex);
        item = buffer.data[buffer.out];
        printf("Consumer %d consumed item %d at index %d\n", (int)arg, item, buffer.out);
```

```
        buffer.out = (buffer.out + 1) % BUFFER_SIZE;
        sem_post(&mutex);
        sem_post(&empty);
    }
}

int main() {
    buffer.in = 0;
    buffer.out = 0;

    sem_init(&mutex, 0, 1);
    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);

    pthread_t prod1, prod2, cons1, cons2;
    pthread_create(&prod1, NULL, producer, (void *)1);
    pthread_create(&prod2, NULL, producer, (void *)2);
    pthread_create(&cons1, NULL, consumer, (void *)1);
    pthread_create(&cons2, NULL, consumer, (void *)2);

    pthread_join(prod1, NULL);
    pthread_join(prod2, NULL);
    pthread_join(cons1, NULL);
    pthread_join(cons2, NULL);

    sem_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);

    return 0;
```

```
}
```

Program 20

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semhore.h>
#include <unistd.h>

#define BUFFER_SIZE 10

struct buffer {
    int data[BUFFER_SIZE];
    int in;
    int out;
};

struct buffer buffer;

sem_t mutex;
sem_t empty;
sem_t full;
sem_t rmutex;
sem_t wmutex;

int rcount;
int wcount;

void *reader(void *arg) {
```

```
    int item;
    int id = *(int *)arg;
    while (1) {
        sem_wait(&rmutex);
        rcount++;
        if (rcount == 1)
            sem_wait(&wmutex);
        sem_post(&rmutex);
        sem_wait(&full);
        sem_wait(&mutex);
        item = buffer.data[buffer.out];
        printf("Reader %d read item %d at index %d\n", id, item, buffer.out);
        buffer.out = (buffer.out + 1) % BUFFER_SIZE;
        sem_post(&mutex);
        sem_post(&empty);
        sem_wait(&rmutex);
        rcount--;
        if (rcount == 0)
            sem_post(&wmutex);
        sem_post(&rmutex);
        sleep(1);
    }
}

void *writer(void *arg) {
    int item;
    int id = *(int *)arg;
    while (1) {
        item = rand() % 100;
        sem_wait(&wmutex);
```

```
        wcount++;
        if (wcount == 1)
            sem_wait(&empty);
        sem_post(&wmutex);
        sem_wait(&mutex);
        buffer.data[buffer.in] = item;
        printf("Writer %d wrote item %d at index %d\n", id, item, buffer.in);
        buffer.in = (buffer.in + 1) % BUFFER_SIZE;
        sem_post(&mutex);
        sem_post(&full);
        sem_wait(&wmutex);
        wcount--;
        if (wcount == 0)
            sem_post(&empty);
        sem_post(&wmutex);
        sleep(1);
    }
}

int main() {
    buffer.in = 0;
    buffer.out = 0;

    sem_init(&mutex, 0, 1);
    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);
    sem_init(&rmutex, 0, 1);
    sem_init(&wmutex, 0, 1);

    rcount = 0;
```

```
    wcount = 0;

    pthread_t r[5], w[5];
    int id[5];
    for (int i = 0; i < 5; i++) {
        id[i] = i + 1;
        pthread_create(&r[i], NULL, reader, &id[i]);
        pthread_create(&w[i], NULL, writer, &id[i]);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(r[i], NULL);
        pthread_join(w[i], NULL);
    }

    sem_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);
    sem_destroy(&rmutex);
    sem_destroy(&wmutex);

    return 0;
}
```

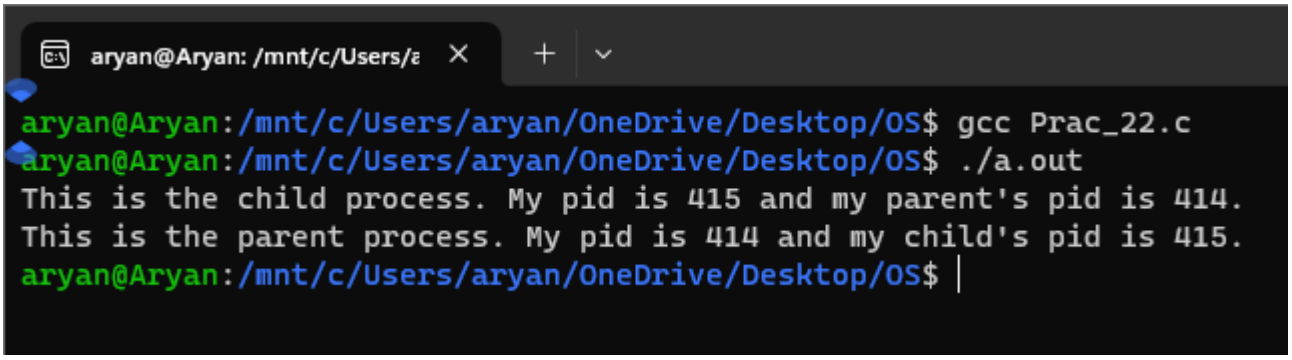
Program 21

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
 pid_t pid;
 pid = fork();
 if(pid==0)
 {
 exit(0);
 }
 else
 {
 sleep(50);
 }
 return(0);
}
```

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
 pid_t pid;
 pid = fork();
 if(pid==0)
```

```
{
sleep(10);
printf("Child Complete");
}
else
{
printf("Parent Complete");
}
return(0);
}
```

Program 22




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

int main() {
    pid_t pid = fork();

    if (pid < 0) {
        perror("fork failed");
        exit(1);
    }

    if (pid == 0) {
        printf("This is the child process. My pid is %d and my parent's pid is %d.\n", getpid(), getppid());
        exit(0);
    }

    else {
        wait(NULL);
```

```
        printf("This is the parent process. My pid is %d and my child's pid is %d.\n", getpid(), pid);
        exit(0);
    }

    return 0;
}
```

Program 23

```
#include<stdio>
#include<unistd>
int main(){
 printf("I am executing A.c \n");
 printf("PID of A.c is %d\n",getpid());
 char *args[]={"./B.c",NULL};
 execv(args[0],args);
}
```

```
#include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>
void main()
{
int n;
printf("\nUp to How Many Numbers:");
scanf("%d",&n);
for(int i=1; i<n; i++)
{
int flag=0;
for(int j=2; j<=i/2; j++)
{
if(i%j==0)
{
flag=1;
break;
}
```

```
}
if(i==1)
{
printf("\n1 is neither Prime nor Composite..\n");
}
else
{
if(flag==0)
{
printf("%d is a Prime Number..\n",i);
}
}
}
}
```

```
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$ gcc A.c -o A
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$ ./A
I am executing A.c
PID of A.c is 481

Up to How Many Numbers:^[[2~

1 is neither Prime nor Composite..
2 is a Prime Number..
3 is a Prime Number..
5 is a Prime Number..
7 is a Prime Number..
11 is a Prime Number..
13 is a Prime Number..
17 is a Prime Number..
19 is a Prime Number..
23 is a Prime Number..
29 is a Prime Number..
31 is a Prime Number..
37 is a Prime Number..
41 is a Prime Number..
43 is a Prime Number..
47 is a Prime Number..
53 is a Prime Number..
59 is a Prime Number..
61 is a Prime Number..
67 is a Prime Number..
71 is a Prime Number..
73 is a Prime Number..
79 is a Prime Number..
83 is a Prime Number..
89 is a Prime Number..
97 is a Prime Number..
```

Program 24

```
#include <stdio.h>
#include <stdlib.h>
int main()
{

int *ptr, *ptr1;
int n, i;
n = 5;
printf("Number of elements: %d\n", n);
ptr = (int*)malloc(n * sizeof(int));
ptr1 = (int*)calloc(n, sizeof(int));
if (ptr == NULL || ptr1 == NULL) {
printf("Memory not allocated.\n");
exit(0);
}
else {
printf("Memory successfully allocated using malloc.\n");
free(ptr);
printf("Malloc Memory successfully freed.\n");
printf("\nMemory successfully allocated using calloc.\n");
free(ptr1);
printf("Calloc Memory successfully freed.\n");
}
return 0;
}
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
int main()
{
int* ptr;
int n, i;
n = 5;
printf("Number of elements: %d\n", n);
ptr = (int*)calloc(n, sizeof(int));
if (ptr == NULL) {
printf("Memory not allocated.\n");
exit(0);
}
else {
printf("Memory successfully allocated using calloc.\n");
for (i = 0; i < n; ++i) {
ptr[i] = i + 1;
}

printf("The elements of the array are: ");
for (i = 0; i < n; ++i) {
printf("%d, ", ptr[i]);
}
n = 10;
printf("\n\nThe new size of the array: %d\n", n);
ptr = realloc(ptr, n * sizeof(int));
printf("Memory successfully re-allocated using realloc.\n");
for (i = 5; i < n; ++i) {
ptr[i] = i + 1;
}
printf("The elements of the array are: ");
```

```
for (i = 0; i < n; ++i) {
printf("%d, ", ptr[i]);
}
free(ptr);
}
return 0;
}

#include <stdio.h>
#include <stdlib.h>
int main()
{
int* ptr;
int n, i;
n = 5;
printf("Number of elements: %d\n", n);
ptr = (int*)calloc(n, sizeof(int));
if (ptr == NULL) {
printf("Memory not allocated.\n");
exit(0);
}
else {
printf("Memory successfully allocated using calloc.\n");
for (i = 0; i < n; ++i) {
ptr[i] = i + 1;
}
printf("The elements of the array are: ");
for (i = 0; i < n; ++i) {
printf("%d, ", ptr[i]);
}
```

```
}
return 0;
}

#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
void main()
{
int n, *ptr, i;
printf("Input array size: ");
scanf("%d",&n);
ptr = (int *)malloc(n*sizeof(int));
if(ptr==NULL)
{
printf("\nNo Allocation of memory");
}
else
{
printf("\nMemory Allocation Done!");
printf("\nAddress of first byte = %p", ptr);
for(i=0; i<n; i++)
{
ptr[i] = i+10;
}
}
printf("\nArray Elements: \n");
for(i=0; i<n; i++)
{
printf("%d ", ptr[i]);
```

}

}

```
Number of elements: 5
Memory successfully allocated using malloc.
Malloc Memory successfully freed.

Memory successfully allocated using calloc.
Calloc Memory successfully freed.
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$
```

```
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$ ./a.out
Input array size: ^[[F^[[2~

Memory Allocation Done!
Address of first byte = 0x55f7d7d1dac0
Array Elements:
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108
109 aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$
```

```
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$ ./a.out
Number of elements: 5
Memory successfully allocated using calloc.
The elements of the array are: 1, 2, 3, 4, 5, aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$
```

```
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$ ./a.out
Number of elements: 5
Memory successfully allocated using calloc.
The elements of the array are: 1, 2, 3, 4, 5,

The new size of the array: 10
Memory successfully re-allocated using realloc.
The elements of the array are: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, aryan@Aryan:/mnt/c/Users/aryan/OneDrive/
Desktop/OS$
```

Program 25

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

int main() {
   for (int i = 0; i < 3; i++) {
    int pid = fork();

    if (pid == 0) {
      int lineNum = i;
      FILE *fp = fopen("courses.txt", "r");
      if (fp == NULL) {
        perror("fopen");
        exit(1);
      }

      fseek(fp, 0, SEEK_SET);
      for (int j = 0; j < lineNum; j++) {
        char line[1024];
        fgets(line, sizeof(line), fp);
      }

      char line[1024];
      fgets(line, sizeof(line), fp);
      fclose(fp);

      printf("%s", line);
```

```
      exit(0);
    }
  }

  for (int i = 0; i < 3; i++) {
    int status;
    wait(&status);
  }
  return 0;
}
```

```
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$ ./a.out
Discrete Structure
Operating System
Java Programming
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$
```

Program 26

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <locale.h>

int main() {
  int n, i, temp;

  // Get the number of integers to generate
  printf("Enter the number of integers to generate: ");
  scanf("%d", &n);

  // Generate n integers
  int numbers[n];
  for (i = 0; i < n; i++) {
    numbers[i] = rand() % 100 + 1;
  }

  // Write the integers to All.txt
  int fdAll = open("All.txt", O_CREAT | O_WRONLY , S_IRWXU);
  if (fdAll < 0) {
    perror("open All.txt");
    exit(1);
  }
  write(fdAll, numbers, n * sizeof(int));
```

```
close(fdAll);

// Read the integers from All.txt
fdAll = open("All.txt", O_RDONLY);
if (fdAll < 0) {
  perror("open All.txt");
  exit(1);
}
read(fdAll, numbers, n * sizeof(int));
close(fdAll);

// Write odd integers to Odd.txt
int fdOdd = open("Odd.txt", O_CREAT | O_WRONLY, S_IRWXU);
if (fdOdd < 0) {
  perror("open Odd.txt");
  exit(1);
}
int oddCount = 0;
for (i = 0; i < n; i++) {
  if (numbers[i] % 2 != 0) {
    write(fdOdd, &numbers[i], sizeof(int));
    oddCount++;
  }
}
close(fdOdd);

// Write even integers to Even.txt
int fdEven = open("Even.txt", O_CREAT | O_WRONLY, S_IRWXU);
if (fdEven < 0) {
  perror("open Even.txt");
```

```
  exit(1);
}
int evenCount = 0;
for (i = 0; i < n; i++) {
  if (numbers[i] % 2 == 0) {
    write(fdEven, &numbers[i], sizeof(int));
    evenCount++;
  }
}
close(fdEven);

// Display the contents of All.txt
printf("\nAll.txt:\n");
fdAll = open("All.txt", O_RDONLY);
if (fdAll < 0) {
  perror("open All.txt");
  exit(1);
}
for (i = 0; i < n; i++) {
  read(fdAll, &numbers[i], sizeof(int));
  printf("%d\n", numbers[i]);
}
close(fdAll);

// Display the contents of Odd.txt
printf("\nOdd.txt:\n");
fdOdd = open("Odd.txt", O_RDONLY);
if (fdOdd < 0) {
  perror("open Odd.txt");
  exit(1);
```

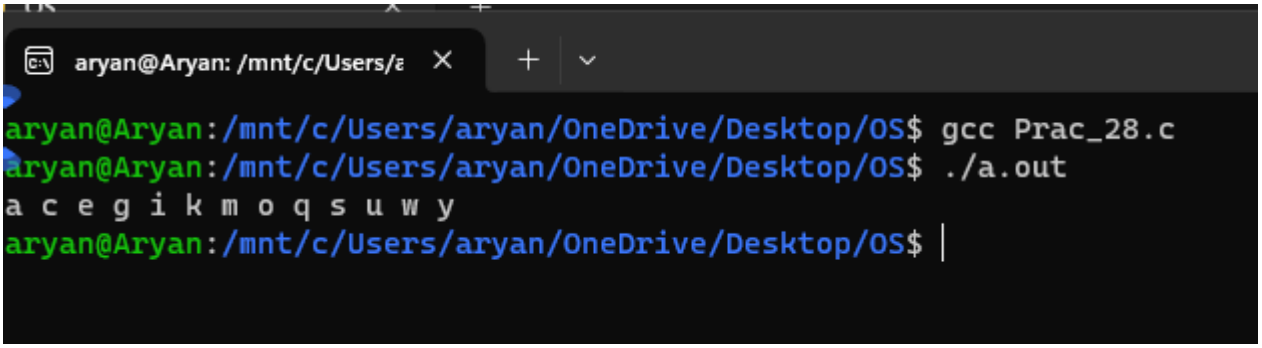
```
  }
  int oddNumbers[oddCount];
  for (i = 0; i < oddCount; i++) {
    read(fdOdd, &oddNumbers[i], sizeof(int));
    printf("%d\n", oddNumbers[i]);
  }
  close(fdOdd);

  // Display the contents of Even.txt
  printf("\nEven.txt:\n");
  fdEven = open("Even.txt", O_RDONLY);
  if (fdEven < 0) {
    perror("open Even.txt");
    exit(1);
  }
  int evenNumbers[evenCount];
  for (i = 0; i < evenCount; i++) {
    read(fdEven, &evenNumbers[i], sizeof(int));
    printf("%d\n", evenNumbers[i]);
  }
  close(fdEven);
}
```

Program 27

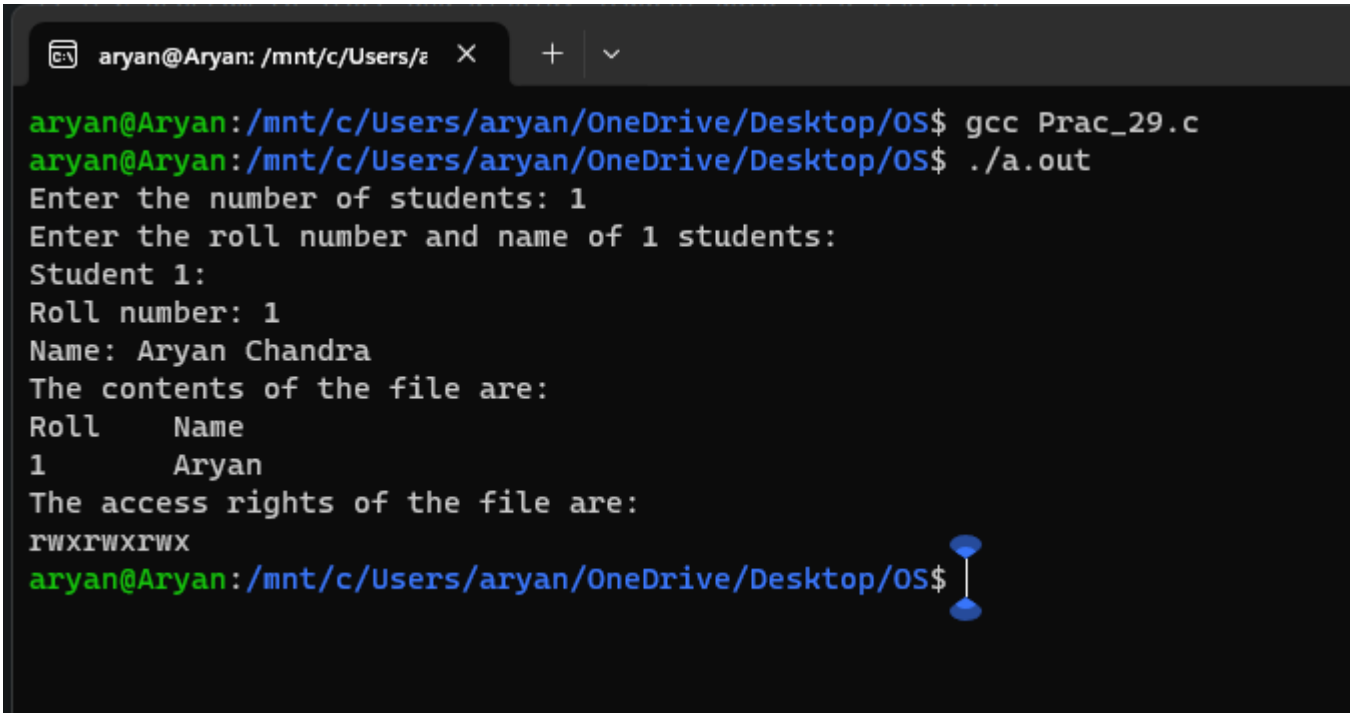
```
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>

int main() {
  char path[1024];
  printf("Enter the file or directory name: ");
  scanf("%s", path);

  mode_t permissions;
  printf("Enter the permission (access rights): ");
  scanf("%o", &permissions);

  if (chmod(path, permissions) == -1) {
    perror("chmod");
    return 1;
  }

  printf("Access rights for %s changed to %o.\n", path, permissions);

  return 0;
}
```

Program 28

```
#include <stdio.h>

int main() {
  char characters[26];

  for (char i = 'a'; i <= 'z'; i++) {
    characters[i - 'a'] = i;
  }

  for (int i = 0; i < 26; i += 2) {
    printf("%c ", characters[i]);
  }
  printf("\n");
  return 0;
}
```



Program 29




```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>

struct student {
    int roll;
    char name[50];
};

void write_data(FILE *fp, int n) {
    struct student s[n];

    printf("Enter the roll number and name of %d students:\n", n);
    for (int i = 0; i < n; i++) {
        printf("Student %d:\n", i + 1);
```

```
        printf("Roll number: ");
        scanf("%d", &s[i].roll);
        printf("Name: ");
        scanf("%s", s[i].name);
    }

    fwrite(s, sizeof(struct student), n, fp);
}

void read_data(FILE *fp) {
    struct student s;

    printf("The contents of the file are:\n");
    printf("Roll\tName\n");
    while (fread(&s, sizeof(struct student), 1, fp) == 1) {
        printf("%d\t%s\n", s.roll, s.name);
    }
}

void display_access_rights(const char *filename) {
    struct stat st;
    if (stat(filename, &st) == -1) {
        perror("stat");
        exit(EXIT_FAILURE);
    }

    printf("The access rights of the file are:\n");
    printf((st.st_mode & S_IRUSR) ? "r" : "-");
    printf((st.st_mode & S_IWUSR) ? "w" : "-");
    printf((st.st_mode & S_IXUSR) ? "x" : "-");
```

```
    printf((st.st_mode & S_IRGRP) ? "r" : "-");
    printf((st.st_mode & S_IWGRP) ? "w" : "-");
    printf((st.st_mode & S_IXGRP) ? "x" : "-");
    printf((st.st_mode & S_IROTH) ? "r" : "-");
    printf((st.st_mode & S_IWOTH) ? "w" : "-");
    printf((st.st_mode & S_IXOTH) ? "x" : "-");
    printf("\n");
}

int main() {
    FILE *fp = fopen("Student.txt", "a");
    if (fp == NULL) {
        perror("fopen");
        exit(EXIT_FAILURE);
    }

    int n;
    printf("Enter the number of students: ");
    scanf("%d", &n);

    write_data(fp, n);

    fclose(fp);

    fp = fopen("Student.txt", "r");
    if (fp == NULL) {
        perror("fopen");
        exit(EXIT_FAILURE);
    }
```

```
    read_data(fp);

    fclose(fp);

    display_access_rights("Student.txt");

    return 0;
}
```

Program 30

```
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <sys/time.h> // Include the header file for 'struct utimbuf'
#include <time.h>
#include<stdio.h>

struct utimbuf {
  time_t acctime;
  time_t modtime;
};

struct utimbuf times = {0, 0};

int main() {

  int fd = open("myfile.txt", O_RDWR);
  if (fd == -1) {
    perror("open");
    return 1;
  }

  off_t offset = lseek(fd, 0, SEEK_SET);
  if (offset == -1) {
    perror("lseek");
    close(fd);
    return 1;
```

```
}

// read-only for the owner, group, and others
chmod("myfile.txt", 0444);

// 0664
umask(0002);

  if (access("myfile.txt", R_OK) == 0) {
  printf("The process has read permission for myfile.txt.\n");
} else {
  printf("The process does not have read permission for myfile.txt.\n");
}



time_t current_time = time(NULL);
times.acctime = current_time;
times.modtime = current_time;

struct utimbuf *times_ptr = &times;

if (utime("myfile.txt", times_ptr) == -1) {
  perror("utime");
  close(fd);
  return 1;
}

close(fd);

return 0;
```

}

```
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$ ./a.out
The process has read permission for myfile.txt.
aryan@Aryan:/mnt/c/Users/aryan/OneDrive/Desktop/OS$
```