OPERATING SYSTEMS DESIGN
PRACTICAL FILE
(CO 204)

NAME- PURAV NAYAK
ROLL NO-2K16/CO/241
BRANCH-CO
BATCH-A4(GROUP G1)
SUBJECT-OPERATING SYSTEMS DESIGN
FACULTY-MS. DIVYASHIKHA SETHIA

INTRODUCTION TO LINUX

Linux is a freely available operating system which was developed by Linus Torvalds with the purpose of having a version of UNIX that was freely available. Linux operating system is an acceptable choice as a workstation and also provides an easy user interface. On the server side as well, it provides database and trading services for companies such as Amazon, US Post Office, German army etc. Linux is an operating system which contains compilers,libraries,development and debugging tools of its own. The C compiler is included for free in the Linux Operating System. Linux Operating System is very versatile and it not only works on workstations,mid and high-end servers, but also on gadgets like PDA's,mobiles,experimental watches etc. Linux Operating System is gradually becoming more and more popular because of it being freely available and through Linux, the Open Source movement can move forward at a very high speed.

LINUX VS UNIX

Linux is great for small to medium sized operations, and today it is also used in place of UNIX in large enterprises. Earlier Linux was considered only as an interesting academic project, but now, with major software vendors porting their applications to Linux, and as it can be freely distributed, the OS has entered the main market as a viable option for Web serving and office applications.

The advent of Linux couldn't prevent UNIX from being the obvious choice in some cases. For example- if an enterprise used massive symmetric multiprocessing systems, or systems with more than eight CPUs, they needed to run UNIX in the past as it was more capable in handling the processes effectively as compared to Linux.

Linux is freely available, as it is an open source OS whereas UNIX is costly when compared to Linux. The midrange UNIX server is priced between 25,000 USD to 249,999 USD. In terms of security and response to the threats to security, both UNIX and Linux are vulnerable to the bugs but Linux is far more responsive in dealing with the bugs as compared to UNIX.

PROGRAM-1

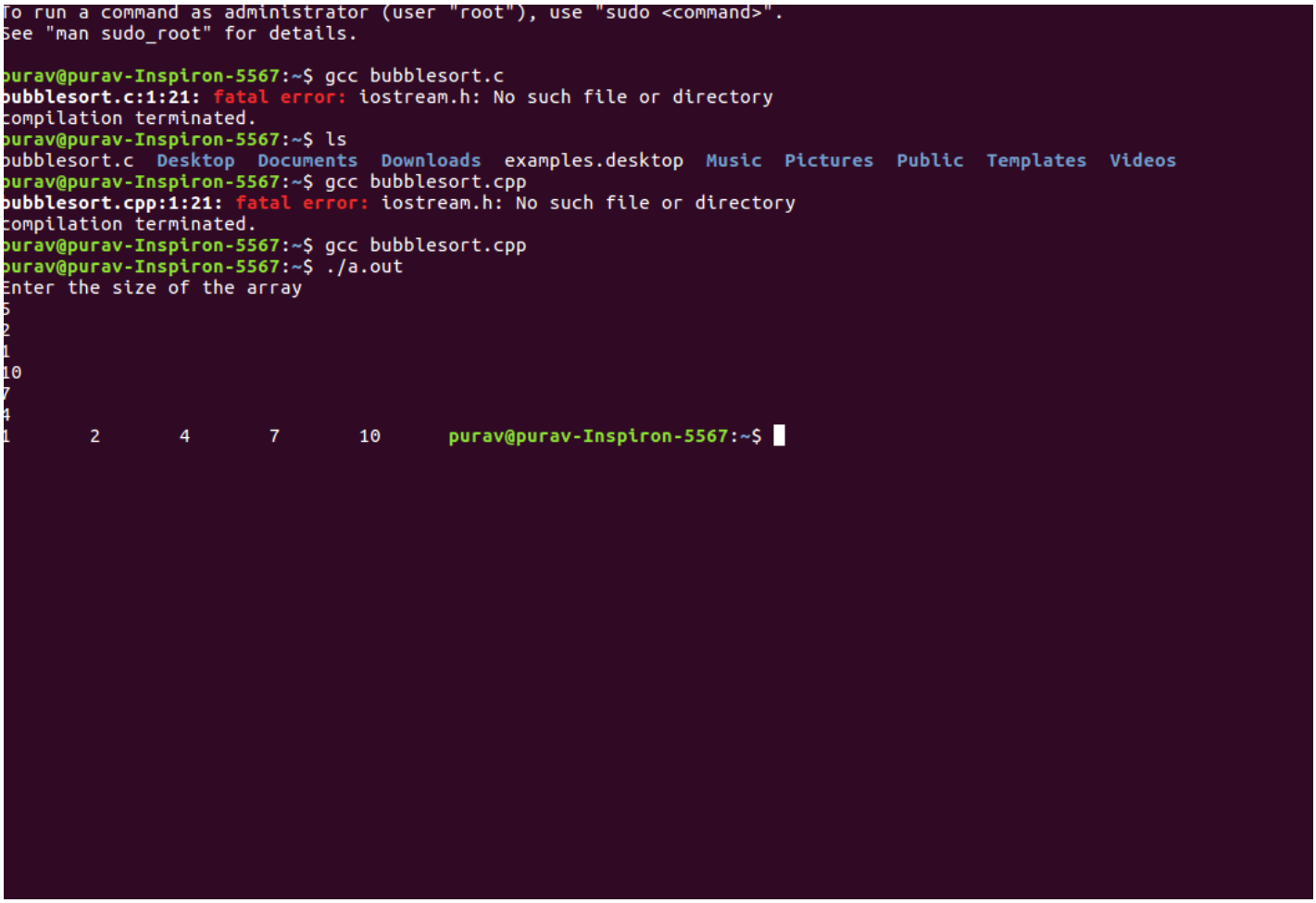
AIM: To write a program for bubble sort in C on linux operating system. Compile it using gcc

INTRODUCTION: Bubble sort is a sorting technique in which each adjacent element is compared and then exchange of elements takes place based on the comparison. To accomplish the compilation of bubble sort in linux operating system, we make use of commands such as gedit,gcc etc.

PROGRAM

```
#include<stdio.h>
int main()
{int i,j,k,l;
int n;
int a[100];
int temp;
printf("Enter the size of the array\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{scanf("%d",&a[i]);
}
for(j=0;j<n;j++)
{for(k=j;k<n;k++)
{if(a[k]>a[k+1])
{temp=a[k];
a[k]=a[k+1];
a[k+1]=temp;
}
}
}
for(l=0;l<n;l++)
{printf("%d\t",a[l]);
}
return 0;
}
```

OUTPUT



LEARNING OUTCOMES

We learnt the basic commands such as gedit,gcc,ls etc and used them to compile and execute bubble sort.

PROGRAM-1

AIM: To write a program for insertion sort in C on linux operating system. Compile it using gcc

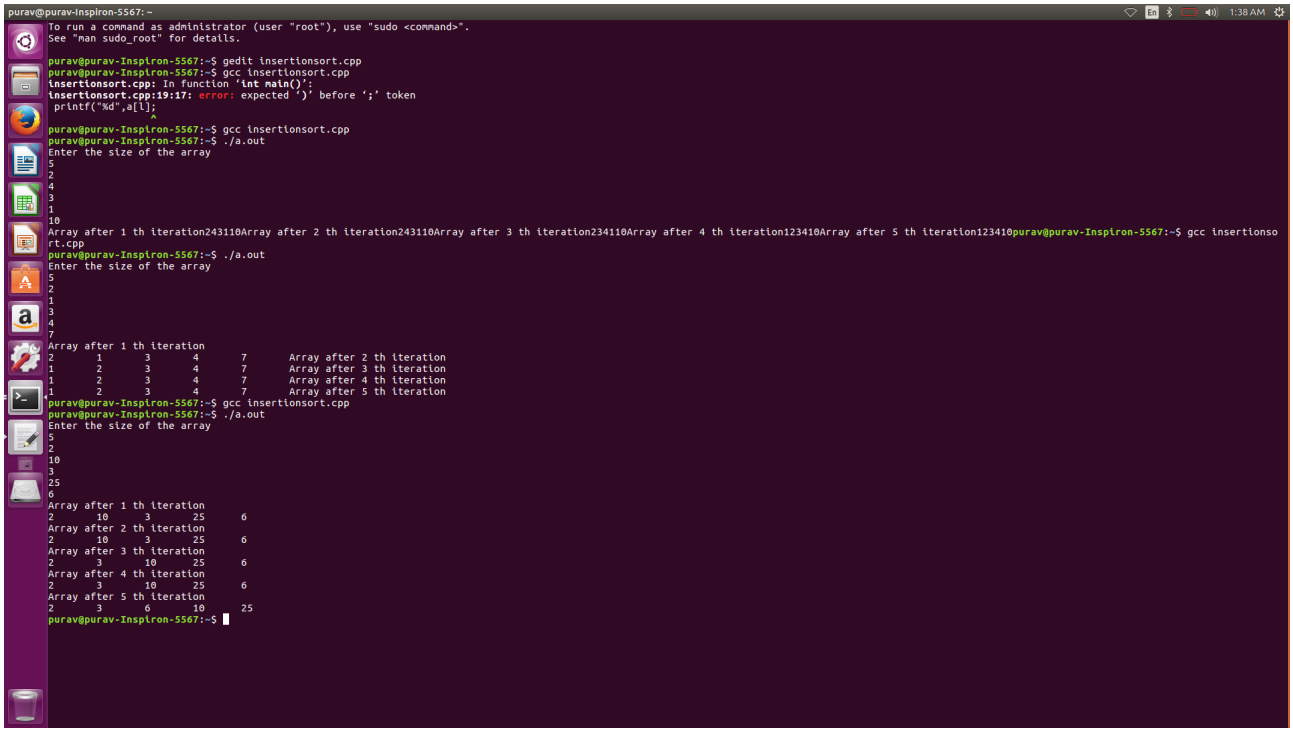
INTRODUCTION

Insertion Sort is a sorting technique in which we take an element as the key and then sort all the elements to the left of the key. We start by taking the first element as the key and then after each iteration, the key is made the next element in the array. We compile and execute Insertion sort on linux operating system by using the basic commands such as gcc,gedit,ls etc.

PROGRAM

```
#include<stdio.h>
int main()
{int i,j,k,l,temp,n,a[100],key;
printf("Enter the size of the array\n");
scanf("%d",&n);
for(int i=0;i<n;i++)
{scanf("%d",&a[i]);
}
for(j=0;j<n;j++)
{key=a[j];
k=j-1;
while(k>=0&&a[k]>key)
{a[k+1]=a[k];
k=k-1;
}
a[k+1]=key;
printf("Array after %d th iteration\n",j+1);
for(l=0;l<n;l++)
printf("%d\t",a[l]);
printf("\n");
}
return 0;
}
```

OUTPUT



LEARNING OUTCOMES

We learnt the basic commands such as gedit,gcc,ls etc and used them to compile and execute insertion sort.

PROGRAM-2

AIM-To write a program to check process ids of the program and parent of the program process

INTRODUCTION-Fork system call creates a new process, which is called child process, which runs concurrently with process (system call fork) and this process is called parent process. After a new child process is created, both processes will execute the next instruction following the fork() system call. A child process uses same pc(program counter), same CPU registers, same open files which are used in parent process. The following program implements forking in C.

PROGRAM

```
#include <stdio.h>

int main(){
        int i;
        for (i = 0; i < 5; ++i){
                if (fork()==0){
                        printf("child %d from parent %d\n",getpid() ,getppid());
                        exit(0);
                }
        }

        for (int i = 0; i < 5; ++i){
                wait(NULL);
        }

}
```

LEARNING OUTCOMES

In this program we learnt to use the fork() function and the concepts of parent process and child process.

PROGRAM-3

AIM: Write two programs serverprogram.c and clientprogram.c and compile them separately. clientprogram must print n numbers and serverprogram must fork a child process which should execute the clientprogram using exec command.

INTRODUCTION: The exec family of functions replaces the current running process with a new process. It can be used to run a C program by using another C program. It comes under the header file unistd.h.
execvp : Using this command, the created child process does not have to run the same program as the parent process does.
The exec type system calls allow a process to run any program files, which include a binary executable or a shell script .
In the following program, we have used the execvp command to fork and execute a child process.

PROGRAM

server.c

```
#include <stdio.h>

void main(int argc,char const *argv[] ){
        int p = fork();

        if(p<0)
                printf("failure to create child\n");
        else if(p==0){
                printf("Executing client.c program in child process %d of
                        parent process %d\n",getpid(),getppid());
                char *args[]={"./client",argv[1],NULL};
                execvp(args[0],&args);
        }

        wait(NULL);

}
```

client.c

```
#include <stdio.h>

int main(int argc, char const *argv[]){
	int  n = atoi(argv[1]);

	for (int i = 0; i < n; ++i){
		printf("%d\t", i+1);
	}

	printf("\n");
	return 0;
}
```

LEARNING OUTCOMES

Through this program, we learnt what the exec command does and how we use it in C to compile an external file while forking a child process.