Write-up	CorrectnessOf Program	DocumentationOf Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 01(A)

Title: Basic Linux commands

Aim: Study of Basic Linux Commands

----

1. **Echo:-** It is used to display line of text listing that are passed as an argumentSyntax:
   echo[options][string]

2. **Ts :-** It is used to display the contents of directorySyntax:
   Ls

3. **read: -** It is used to read the contents of a line into VariableSyntax:
   read [otions] [hone]

4. **Cat:-** It is used to display the content of a fileSyntax:-
   cat <Filename>

5. **touch:** It is used to create emply files
   Syntax
   touch <Filename>
   touch<File1> <Filer >

6. **grep:** It filters the content of a file which makes our search easySyntax:
   Command grep <Search Word>

7. **sed:** It is used to edit streams using regular expression but this editing is not permanent Itremains only in display but In actual file content remains same

    Syntax:

    Sed[options]… {Script-only -if other script}[Input file]

8. **pwd:** It is used to display the location of current working directorySyntax:

    Pwd

9. mkdir: It is used to display create a new directory under only
directorySyntax:
mkdir<directory name>

10. rmdir: rmdir is used to resume a
directorySyntax:
rmdir<directory name>

11. cd:It is used to change the
directorySyntax:
Cd <directory name >

12. rm: It is used to display remove the
fileSyntax:
rm<file name ><directory file>

13. mr: It is used to move a file directory from one to
anotherSyntax:
Mr<filename><directory path>

14. rename: It is used to resume a
fileSyntax:
Rename/old name /new name/file

15. user add: It is used to add remove a user an linux
serverSyntax:
User add username

16. sleep: It is used to hold the terminal by the specified amount of time by default it takes time inseconds
Syntax:
Sleep<time>

Write-up	CorrectnessOf Program	DocumentationOf Program	Viva	Timely Completion	Total	Dated Sign ofSubject Teacher
2	2	2	2	2	10	

Assignment No. 01(B)

Title: Program to important on address bank

Aim: Write a program to implement on address book

Q. Write a program to implement on address bade with options given

below.a] create address book

b] view address

bookc] insert a

record

d] delete a record

e] modify a

recordf] exit

***Introduction** – Shell programming is a basic skill that unix system admin has that systemadmin must be able to write shell programming because there are quickly authomated by using shell programs many taste that can be quickly authomated by using shell programs.

Q. What is shell programming ?

It is a test that continous standar unix & shell command. Shell has a programming support such as - Comment variable, conditional commands reputed action command. Shell program is on interpreted program also known as shell script or batch file of upis commands like BATFiles on windows command lines.

- Create a shell program

i] Create a text file to han hold the shell

programii] Decide which shell you use

iii] Add the required commands to file save the files

iv] Change the permission on file so that it is

executable.v] Run the shell program

Program -

```
 Create()
{
echo "Enter No. of records want to
enter.”   read m
while [$n! =0]
do
echo "use ID, user name, user address, user mobile
number"read v-data
echo $ v-data >> adr,
txtn = $.((&n-1))
done
}
Search()
echo "Enter names want to
search"read $name,
res = $ ( grep -co " $hname "adr.
txt)if[$2 –   eq1]; then
echo "No match
Found"else
echo
"$res"f;
}
det()
{
echo "Enter name to want to
delete"read name
```

```
res = $(grep-co" $name" adr.

txt)if [$?- eq1]; then
```

```
echo "No match
Found"else
grep - v-co" $ name" adr.txt>
temp.txtcp.temp.txt adr txt
f;
}
modify()
{
echa "Enter name you want to modify"
read name
echo “   Enter modified
name"read new - nm
sed-i "s/$name/$new –   w/" adr text
}
while
truedo
echo "Enter choice.
"read ch
case #ch
in1]
create
2] cat adr.txt
3] Search ;;
4] del;;
5] modify;;
6]       e
xit;;
Esac
```

done

output -

Enter choice

 1) create

```
 2) show record
 3) Search
 4) delete
5) modify
6) exit

1) Create
Enter number & records wants to enter
User ID, Username, user address, user mobile
 numberOL                 ABC          BOM
              1234567890
User ID: username, user address, user mobile
number02                  DEF          Nashik
            9503867142
   Enter
   choice1…
   6
     6
2] Show record

01 ABC BOM 1234567890
02 DEF Nashik 9503867142

Enter
choice1…
 ……. 6
3)      Enter name want to
searchDEF
```

```
02 DEF Nashik 9503867142

Enter

choice1… 6

4]      Enter name you want to

deleteDEF

Enter choice
```

1… 6

5] Enter name. to

modifyABC

Enter modified

nameXYZ

Enter

choice1… 6

6

–––––––––––––––––––______––––––––––––––––––––––––––––

Conclusion -Thus, we learn about shell programming & shell realeted command & itsimplementation.

–––

Write-up	Correctnes sOf Program	Documentatio nOf Program	Viva	Timely Completio n	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 02(A)

Title: Process Control system call.

Aim: The Demonstration of fork, execu and wait system call along with zomile and orphan state

----

Theory:

Fork:

It is a system call that creates a new process under the UNIX operating system. It takes no arguments. The purpose of fork () is to create a new process, which becomes the child process of the caller. After a new child process is created, both processes will execute the next instruction following the fork () system call. Therefore, we have to distinguish the parent from the child. This can be done by testing the returned value of fork ():

If fork () returns a negative value, the creation of a child process was unsuccessful.

fork () returns a zero to the newly created child process.

fork () returns a positive value, the process ID of the child process, to the parent. It isassociated with each job as unit of time to complete.

The returned process ID is of type pit defined in sys/types.h. Normally, the process ID is an integer. Moreover, a process can use function getpid () to retrieve the process ID assigned to this process. Therefore, after the system call to fork (), a simple test can tell which process is the child.

Therefore, the parent and child processes have separate address

spaces.Let us take an example:

```
int main ()
{
        printf ("   Before
        Forkingfork ();
        printf ("   After Forking"   ); return 0;
}
```

If the call to fork() is executed successfully, Unix will

Make two identical copies of address spaces, one for the parent and the other for the child.

Both processes will start their execution at the next statement following the

fork() call.If we run this program, we might see the following on the screen:

```
Before Forking

After Forking

After Forking
```

Therefore. Here, printf () statement after fork () system call executed by parent as well as childprocess.

Both processes start their execution right after the system call fork (). Since both processes have identical but separate address spaces, those variables initialized before the fork () call have the same values in both address spaces. Since every process has its own address space, any modifications will be independent of the others. In other words, if the parent changes the value of its variable, the modification will only affect the variable in the parent process's address space.

Consider one example which distinguish the parent from the

child#include <stdio.h>

```
#iclude style h>

    void    childprocess

    ()                  void

    parentprocess ()

int main ()

{

    pid - + pid;

    pid =

    fork(c);If
```

```
        (pid == 0){

                childprocess ()

}

else {

        parent process

         ():return 0;

}
```

```
void childprocess (){
        void parenprocess
        ()
}
```

In this program, both processes print lines at Indicate (; whether, the line is printed by the child by parent (2) the Value of von oble, when the main program executes Park.0, O identical copy of address pace including the program & all data is created system call fi reloan the child process the following shows that in bath address space there is a portable ptd. There in parent reverse the childprocess ID GUES & the one in the child reserves.

parent	child

Main () {
Pid = fork
()
If (pid == 0){
Childprocess
();
}
Void childprocess (){}
Void parentprocess (){4}
Void parentprocess (){4}
	Main () {
	Pid = fork
	()If (pid ==
	0)
	Childprocess ();
	}
	else{
	Parentprocess();
	}
	Void childprocess (){}
	Void parentprocess (){4}
	Void parentprocess (){4}

Now, both the program will executed independent of each other after string at the next statement.

Parent	child

Main () pid 456
{Pid = fork ()
If (pid == 0){
Childprocess
();
}
Void childprocess (){4}
Void parentprocess (){4}
	Main () pid= {
	Pid = fork ()
	If (pid == 0)
	Childprocess ();
	}
	else{
	Parentprocess();

	}
	Void childprocess (){}
	Void parentprocess
	(){4}

In the parent, since pid is non-zero at calls function parentprocess (). Due to that cpu schedular will design a time quantum to each process the parent or the child process will run for the same time, before the control is switched to other. AND the running process will print some line before you can see only line printed by other process

Ps Command It is used to display /view information related to a process sunning in a Linux system

By default, the ps program shown only process that maintain a connection with a terminals other process run without needing to communicate, with a uses to managed shared resources, we can use ps to see all such process using 'e option 4 to get Bull information with ' &'

Conclusion: - Thus, we're studied process control system calls The demonstrate of the fork ()system call execution & wait system calls along with zombie &.orphan system

Write-up	Correctnes sOf Program	Documentatio nOf Program	Viva	Timely Completio n	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 02(B)

Title: Fork and execv system call

Aim: Demonstration on fork and execv system call

Theory

Fork

–

```
#include

<Systype.b>

#include < unisid.h>

pid - t Pork (void);
```

When the process makes the fork system call, a new process is created which is the done of thecalling process the code, data, stack of new process, called the child process

The newly created process is called child process, whereas calling process is called as parent process, However, there is a difference between the parent & child process, the return value of Pork in child process: 0, whereas in the parent process, the process. ID of child process is standard. Indeed the two process. the difference to figure out whether they are parent or child.

Exec V:

```
#includes <unistd.h>

 int exec c const clear * filename, char *count argv[] char*count envp()
```

execv program is are example where the difference between o 4 a process cannot be emphasized enough process is on created by kernel to execute a program, written to do some tasks. argu() is an array of argument to the program where the 0th element in the away is the Filename of program itself.

execv does not return on success. It can't segment has been initialized From the new program being executed and return address. However, if exev is unsuccessful, -1 is returned & exit 0 is setaccordingly

exec family -

exec() &

execlp()

- **execl()-** Replaces the current process with new process specified by path No.
return is madebecause the calling. process is replaced by new process image eg.
exec("bin\","$""-1" Nuu;

- **execp()** - It gives the user option to specify the Filename & program is searched in directories that are listed the current PATH environment variable. If filename contain slash, its treated relatives or absolute pathname eg execp ('k, "Is"," "-1" Null);

execv() & execup (): - execy () - Executes the file named. by filename as new process. The argument is an array of null - received string that is used to provide a value for the Begy argument to main Punction of the program executed. The last element of this amay must be null pointer. eg. char + argv[] = ("JS","-1", Null);

expcvp():- The created child process doesn't leave to scan the same program as the parent process does. Thus a Fully qualified name would not leaved.

Conclusion - Thus, we've studied the demonstration of Fook & exec system calls

Write-up	Correctnes sOf Program	Documentatio nOf Program	Viva	Timely Completio n	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 03(A)

Title: CPU scheduling Algorithm on Shortest job first

Aim: Implement the C program for CPU scheduling algorithm on shortest job first.

Q. What is Shortest Job first scheduling?

It is the algorithm in which the process having shortest execution time is close for next execution. This scheduling method can be primitive or non-primitive. It significantly reduces the arrays waiting time per other processes awaiting execution.

Types of SJF Methods:

There are 2 types of SJF: 1) Non-Primitive SJG

2) Primitive SJG

It is associated with each job as unit of time to complete.
This algorithm is helpful for batch type pre-casting, where waiting per jobs to complete isnot critical.
It can improve process throughout by making sure that shorter jobs are executed first,hence possibly turnaround time.

1. **Non– Primitive SJF** - In this scheduling, once the CPU cycle is allocated to process, theprocess holds it till, it reaches a waiting state or terminated.

2. **Primitive SJF -** In this scheduling, jobs are put it into ready queue as they come. Aprocess with shortest burst time being execution

Advantages of using SJF method:

1. It is Frequently used for long term scheduling.
2. It reduces average waiting time over FIFO algorithm.
3. SIF method give lowest average waiting time for Specific set of process.

Disadvantages of using SJF method:

1. Job Completion time must be known earlier, but it is hard to predict.
2. It is often used in a batch system for long term scheduling.
3. It is an algorithm in which the process having the smallest execution line is close for thenext execution.
4. It is associated with each job as a unit of time to complete.
5. It is algorithm method is helpful for back type processing, where waiting for job tocomplete is not critical.

Conclusion: - Thus, we've studied SIF algorithm & its two types.

Write-up	Correctnes sOf Program	Documentatio nOf Program	Viva	Timely Completio n	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 03(B)

Title: CPU Algorithm Round Robin scheduling Algorithm

Aim: Implement the C program for CPU algorithm Round Robin scheduling Algorithm.

Round Robin Scheduling Algorithm:

This algorithm is used to schedule the process fairly for each Job a time slot / quantum / interrupting the job. It is not completed then, the job come after the other job, which is arrived in the quantum time that makes these scheduling fairly.

Round Robin is cyclic in nature, so starvation doesn't occurs It is a variant of FCFS scheduling. No priority, special importance is given to array process or task. RR scheduling is also known as time slicing scheduling.

We get three time which are: 1. Completion time

2. Turnaround time

3. Waiting time

1. Completion Time: The time taken for process to complete.

2. Turnaround time: Total time the process exists in the system.

3. Waiting Time: Total time waiting for their complete execution

Explanation of Program:

we ask to user to outer the no. of process arrival & lowest time for each process. We calculate waiting time 4 process turnaround time using RR algorithm. The main port is to calculate wanting & turnaround time TA time is calculated by adding total time taken & subtracting the arrival and burst time from total and adding to waiting time. This is how PR scheduling take place.

Conclusion: Thus, we've studied Round Robin Algorithm

Write-up	Correctnes sOf Program	Documentatio nOf Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 04(A)

Title: Thread Synchronization using counting semaphores.

Aim: Application to demonstrate: Producer – Consumer problem with counting & Mutex

---

Theory: The producer /Consumer Problem

The general statement is this, there are our of more producer generating come type of data & placing there in buffer. There is a single consuming that is taking stems out of buffer at a time. The system is to be constrained to prevent the overlap may accept the buffer at any time. The problem is to make sure that the producer wants to try to add data into the buffer

Solution to bounded /circular buffer producer consumer problem

```
b[1]    b[2]    b[3]    b[4]    b[5] .............  b[n]
         out
b[1]    b[2]    b[3]    b[4]    b[5] .............  b[n]
                 In      (b)     Out
```

```
 Producer
while (true)
{
  /*producer item*/
  b[in] = v;
```

```
Consumer
 while true
 {
   while(in<=out)
   /* do nothing */;
```

```
    int:
}
```

```
  w = b[out]
  out : t
  /* consumer item */;
}
```

Solution using semaphore (Unbounded buffer)

```
Semaphore S=1,
n=ovoid producer
()
{
 while (true) {
 producer();
 Sem.wait
 (3);
 Append();
 Sem.signal (s);
 Sem. Signal
 (n),
}}
Void consumer()
{while (true) {
Sem.wait (n);
Sem.wait (s)
take ();
sem. Signal
(3);
consumer();
}}
Void () {
parbegin (producer, consumer);
}
```

Semaphore:

It is a special type of variable containing integer value used for signalling processes Only 3 operations. may be performed on a semaphore, all of which are atomic initialized decrement & increment. The decrement operation may result in unblocking of a process, also known as counting semaphore or a general semaphore. A semaphore may be initialized to a non-negative integer value The semwait operation, the semaphore value. It is the value becomes negative thenthe process executing the semwait is blacking otherwise the process continue.

Process thread P-Semaphore Process thread

of execution. Operation of execution

Non-Critical Section Critical section Non-Critical

Sectionof process A of process B

V-Semaphor

eOperation

Generalized use of semaphore for forcing critical

section,Semaphore SV=1;

```
loop forever
{
  wait (sv);
  critical
  codesection;
  Signal(sv);
  non- critical codesection;
```

Conclusion: We've studied thread synchronization using counting & procedure consumerproblem with counting semaphore & Mutex.

Write-up	Correctnes sOf Program	Documentatio nOf Program	Viva	Timely Completio n	Total	Dated Sign of Subject Teacher
2	2	2	2	2	10	

Assignment No. 04(B)

Title: Thread Synchronization & Mutual execution Mutex

Aim: Application to demonstrate: Reader - Writer problem with reader priority.

Introduction:

The reader writer problem is our of the classic Synchronization problems like the dining philosophers. It often used to constant synchronization mechanism. It is also on initially practicalproblem.

Consider a situation where we’ ve to file shared bet many people.

It one of people tries to edit the file, no other people. should be reading or writing at thesame time, otherwise the changes will not be visible to him/her.
However, Same person is reading the File, then other may read it at some time.
This is called reader-writer problem.

Problem Statement:

One set of data is shared among a no of processes.
Once, a winter is ready it performs write operation at time.
If a process is writing no other process can read it.
Reader count write, but only reads.

Function of Semaphore:

1. Wait() :- Decrements the semaphore value.
2. Signal() :- Increments the semaphore value.

Writer Process:

1. Writer request entry to critical section.
2. If wait() gives true value, it enters and performs the write. If not allowed, it keeps onwaiting.
3. It exits the critical section

```
do {

// writer request for critical section wait (wrt);

// performs the write

// leaves the critical

section.Signal (wrt);

}  write (true);
```

Reader's Process:

1. Reader requests the entry to critical section.
2. If allowed:

    It increments the count of no. of reader inside the critical section. If this reader is First reader is entering, then it locks the writes semaphore to restrict the entry of writer.

    If then signal mutex, as any other render is allowed to enter while other are already reading.

    After performing reader exits the critical section, when existing, it checks if no more reader is inside. It Signals the semaphore "wit" os a new writer can outer thecritical section.

```
do {

//reader wants to enter the critical

section.wait (mutex);

read cnt ++;

if(read
```

cnt==1)wait

(wrt); signal

(mutex);read

cnt–;

```
if (read cnt ==

0)signal (wrt);

signal (mutex);

} while (true);
```

Conclusion:

We've studied thread synchronization & mutual exclusion using mutex & diningphilosophers.

Write Up	Correctnes sof Program	Documentatio nof Program	Viva	Timely Completio n	Total	Sign of Subject Teache r
2	2	2	2	2	10	

Assignment No. 05

Title: Deadlock Avoidance Algorithm

Aim: Implement the C program for deadlock avoidance algorithm: Bankets algorithm.

Theory: Deadlock is a situation, where two or more completing actions are waiting for other to finish, thus neither ever does. When a new process enters the system, it must be declared max number of instances of each resources type it needed. This number may exceed the. total number of resources in systems. When a user requests the set of resources, the system must determine whether the allocation of each resource will leave the system in safe state. It' ll allocateresources,otherwise some process must wait until some other process release the resources.

Data Structure

N=Total number of resources

H = Total number of resources types

Available[j]=k, k = instance of resource type Rj available

If max[i , j]=k, Pi may request at most " k" instance resources

Allocation = If allocation [i,j]=k, " Pi" allocated to instance " k" of Rj

resource.If need [i,j]=k, " Pi" may need more instances " k" of

resource type Rj,

need [i,l] = max[i,j] - allocation[i,j]

Safety Algorithm

1) Work and finish be vectors of length ‘ m’ and

‘ n’ respectively.Initialize: Work = Available

Finish [i] = false, for i=1,2,3, 4… .., n

2) Find and ‘ i’ such that both

A) Finish [i]=false

B) Need i <= Work

If no such I exist, go to step 4

3) Work = Work +

Allocation [i]Finish[i] = true

Go to step 2

4) If finish [i] = true for

all iThen the system is in safe

state

Resource-Request Algorithm

1) If Request’ i’ <= Need‘ i’

Go to step 2, otherwise raise an error condition, since the process has exceeded itsmaximum condition.

2) If Request’ i’ <= Available,

Go to step 3, otherwise Pi must wait, since the resource not available.

3) Have the system pretend to allocated request resource to process Pi by modifying thestate as follows:
    a. Available = Available – Request‘ i’
    b. Allocation’ i’ = Allocation‘ i’ + Request‘ i’
    c. Need‘ i’ = Need’ i’ - Request‘ i’

If the resulting resource allocation state is safe, the transactions is completed and process Pi isallocated its resources. However, if the state is unsafe, the Pi must wait for request, if old resource allocation state is restored.

Algorithm

1) Start
2) Get the value of process and resources.
3) Get the available value.
4) After allocation find the need value.
5) Check whether if it is possible to allocate.
6) If it is possible, then the system is in safe state.
7) Else, system is not in safe state.

8) If new request comes, check the system is in safe state.
9) If it is not in safe state, do not allow the request.
10) Stop.

Conclusion: We’ ve studied Deadlock Avoidance algorithm using Banker’ s Algorithm.

Write Up	Correctnes sof Program	Documentatio nof Program	Viva	Timely Completio n	Total	Sign of Subject Teache r
2	2	2	2	2	10	

Assignment No. 06

Title: Write a C program for Page Replacement Algorithm FCFS, LRU & Optimal for frame sizeas minimum tree.

Aim: To implement C program for Page Replacement Algorithm FCFS, LRU & Optimal forframe size as minimum tree.

Theory: 1) LRU Page Replacement Algorithm

In OS, that use paging for memory management, page replacement algorithm is needed, which page needed to be replaced, when new pages come in. Whenever a new page is referred, and not present in memory, page fault occurs, and OS replaces existing page with newly added page.

Different page replacement algorithm suggests different ways to decide which page to replace. The target is for all algorithm is to reduce the number of page faults. In LRU, it is greedy algorithm, where the page is to be replaced, that least used recently. The idea is based on localityof reference, the least recently used page is not likely.

Let say the page reference string 7012030432023.

Initially, we’ ve 4 pages slot empty. Initially, all slots are empty. So, when 7012 isallocated to empty slots => 4 pages faults.

0 is already there, so there are 4 pages faults.

When 3 come, it’ ll replace 7, it is least recently used, so it has 1-page

faults.0 is already in memory, so 0-page faults.

4 will take place, so 1 page fault.

Now for the further page reference string, so it has 0-page faults, because they’ re already inmemory.

7	(0)	7	(3)	3	(0)	3	(4)	3	
0	→	0	→	0	→	0	→	0	→.......
1		1		1		1		4	

```
2          2          2          2          2
```

Total page faults = 4

2) Optimal Page Replacement Algorithm

In OS, whenever a new page is referred and not present in memory, page fault occurs & OS replaces existing page with newly added pages. Different page algorithm for replacementsuggests different ways to decide which page to replace. The target for all algorithm is to reducenumber of page faults.

In this algorithm, OS replaces the page that will not be used for longest period of time in future.

I) If the referred page is already present, increment hit count.
II) If not present, find in a page that is never referred in future. If such a page exists, replacewith a new page.
III) If no such page exists, find a page that is referenced further in future & replace it withnew page.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2		2			2		2				1			
	0	0	0	0	0		4			0		0				0			
		1	1	1	3		3			3		1				1			

3) FCFS Page Replacement

A FIFO replacement algorithm associates with page the time, when that page was brought into memory. When a page must be replaced, the oldest page is chosen, we can create a FIFO queue to hold all page in memory, we replace the head out of queue. When page is broughtinto memory, we insert it at the tail of queue.

Example: We’ re taking 12-page reference string:

4, 7, 6, 1, 7, 6, 1, 2, 7, 2, 7.

Also, we’ re considering the capacity of queue is 3. Initially, out of queue is empty.

4	7	6	1	7	6
4	4	4	7	7	7

```
7       7       6       6       6
6       1       1       1
```

PF=1	PF=2	PF=3	PF=4	PF=4	PF=4

1	2	7	2	7	1
7	6	1	1	1	1
6	1	2	2	2	2
1	2	7	7	7	7
PF=4	PF=5	PF=6	PF=6	PF=6	PF=6

Conclusion: Thus, we’ ve studied different page replacement algorithm, LRU, FCFS & OptimalPage Replacement

Write Up	Correctnes sof Program	Documentatio nof Program	Viva	Timely Completio n	Total	Sign of Subject Teache r
2	2	2	2	2	10	

Assignment No. 07(A)

Title: Inter process communication in linux using following

Aim: A)FIFOS : Full duplex communication between two independent process first process concept sentences and write on one pipe to be ready by second process and second process count the number of characters, number of words and number of lines in accepted sentence writes this output in a text file and write the contents ofthe file on second pipe to read my first process and displays on standard output.

Theory :

Pipes and FIFOS : Pipe is a mechanism for interprocess communication data returned to the pipe by one process can be read by another process the data is handled in a first is first.out.order the pipe has no name it is created for for one use and both ends must be inherited from the single process which created the pipes.

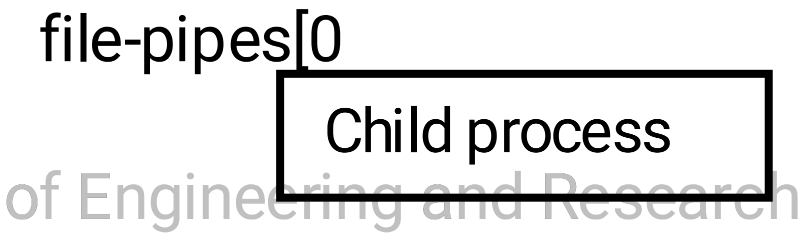
A FIFO special file is similar to pipe but intend being and anonymous temporary connection a fifo has a name or names like any other file process open the FIFO for my name in order to communicate through it a pipe or FIR has to open at both ends simultaneously it execute it as a subprocess.

Neither pipe not FIFO per special file unlock file position in both reading and writing operations happens sequentially reading from the file and writing at the end.

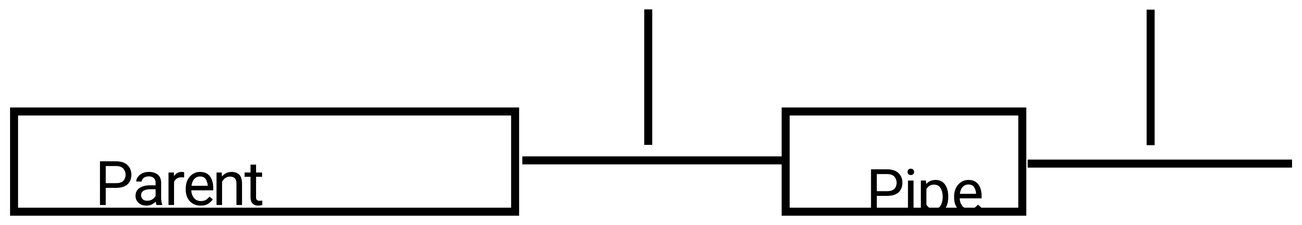
The Pipe Call : The lower pipe() function provides a means of pocessing data between two programs without the overload of in a shell to interrupt the requested command it also give us more contract over the reading and writing of thedata.

The FIFO function has the following

prototype :#include<unistd.h>
int pipe cint file,description[2]);
file.pipes[1]



]



Pipe to a sub process : A common use of pipes is to send data to receive data from a program being run as a subprocess one way of doing this is by using acombination ofpipe fork.

Function : FIRE*popen (c const chat*command.const chat * mode)

This if popen function is closed related to the system function see running and commands.
if the event of error popen and return a null pointer this might happen if the pipe or a system/stream cannot be executed.

Function : int p close(FIFF * stream)

The pclose function is used to close a stream created by popen it waits for the child process to terminate and returns its value as for the system function.

FIFO special files :A special file is a similar to a pipe expect that is its created in a different way instead file is entered into the file system by calling mkfifo,the mkfifo for function is select cleared in the header file system sysl state.h function int mkfifo const char
* filename mode – fmode)

The normal successful return value from mkfile is 0. in the scope of an error conditionare defined for this function.
E exist : the name file already exist.
E nosfc : the direct or file cannot be extended
E ROFL :the direct that would contain the file a side on a read-only file system

Conclusion : In this practicle we studied the full duplex communication between independent processes also performed the operation to see the results and studiedthe FIFO system.

Write Up	Correctnes sof Program	Documentatio nof Program	Viva	Timely Completio n	Total	Sign of Subject Teache r
2	2	2	2	2	10	

Assignment No. 07(B)

Title: Inter-process communication using shaled memory using system v.

Aim: Applications to demonstrate: Client server program in which server process create a staredmemory segment and writes the message to the shared memory segment client process reads themessage from the shared memory segment anddisplay it to the screen.

Theory :

Signals : The common communication channel between user space program and kernel is given by the system calls but there is a different channel that we that of the signal use between user process and to user process.

Sending signals:A process program signal different program using the kills systemcall to user process

Int kill(pidt.pid.int sig);

this will send with the number of sig0 to the process with the process ID PID signal numbers are small positive integer user can send a signal from the command line using the kill command.Common user are kill in to kill the process with pict N or kill in to force processN to read its configuration file certain user actions will make the kernel send a signal to that process for an illegal instruction one gets signal for accessing the science is EGV

Receiving signals :When a process receive a signal a default action happens untess the process has arranged to handle the given signal for the list of signals and corresponding default access traditionally one sets up handler for the signal system call with prototype

```
Typedef          void
                 (*sighandler.t)(int)
Sighandler.tsignal(int sig.signal handler +
```

handler);

Semantics : The traditional semantics was reset signal behaviour to SIG DFL upon innovations of the signal handler possible this was done to avoid the recursive innovations the signal handler is involved with the signal handler returned the interrupted activity is continued.

Blocking signals : Each process has least of concurrently block signals when a signal to blocked it is not delivered remains pending this sigpromst() system call serves to change the list of block signals sigpending() system all reveals what signals are pending.

Weight and BIGCWID :

When a process fork off a child to perform some task it is probably be inserted in is in the things must upon exist is the child lives and exit.The child leaves and exit status that should be returned to the parents so when the child finishes it becomes a zombie. A process that dead already but does not disappears yet because it has not it respected its exit status.

Whenever something interesting happens to the child and in particular when it diesthe parent send a sigchild signal it the parent is not interested it can say so explicitly using.

```
Signal[sighchild-sig-
IGN];Or
Struct.sig.action.act;
act
sa.handler=something.
act.so.flags=SA
NOCIOWATT;
sigaction(sigchild & act,Null);
```

It depends on the Unix flavour whatever SIGHCHILD is sent when SA NOCLDSTOP was set,SA-flag=SA NOCLDSTOP was set. After act SA-flag is sent when children star or stopped children continue.If the parent exist before the child then the is represented to in it process & this process will read its status.

Conclusion : In this praticle we learned about inter process communication using system v and implemented client server program in which server creates issued in memory.

Write Up	Correctnes sof Program	Documentatio nof Program	Viva	Timely Completio n	Total	Sign of Subject Teache r
2	2	2	2	2	10	

Assignment No. 08

Title: Disk Scheduling Algorithm

Aim: Implement C program for Disk scheduling algorithm, SSTF, SCAN, C-Look, consideringthe initial head position moving away from the spindle.

– .

Theory:

Introduction:

A file is a collection of related information that is recorded and secondary storage or file is a collectionof logically related entities from users perfect perspective a file is the smallest allotment of secondary storage

The name of file is divided into two parts1.name
2. Extension separated by a raiderFile attributes and operationsFind directories:
Collection of files is a file directory in directory contains information about the file including attributes location and owner see much of this information is specially that is concern which storage is managed by the operations
16 directory is itself a file accessible by various file management fruitsInformation contained on device directory r:
Name type address current length maximum length date last updated date last accepted owner IDprotection information
Operation performed on directories are:
Search for a file create a file delete a file list a directory rename a file drivers the file system

Advantages of maintaining directories: Epcn a file can be located more quickly

Naming it becomes convenience for users as to users can have same name for different files as mayhavedifferent name for same file
Grouping logical grouping of file can be done by
propertiesFile sharing:
File sharing is the practice of sharing or offering access to digital information resources includingdocuments multimedia graphics computer programs image and ebook
It is private or public distribution of data or resource in a network with different levels of sharingprivileges
File sharing can be done using survey methods the most common techniques for file storage distributedand transmission include the following
Removal storage devices
Centralised file hosting server installation on
networkDistributed peer to peer network

File structure: file system provides sufficient access to the disk by allowing data to be stored talk it did and it arrived in a conveyance nearest way a file system must be able to store the file locate the file and retrieve the profile most of operating system used layering approach for every dog including file systemsevery layer of the file system is responsiblefor some activities

Different layers of file:

When an application program ask for a file the first request is directed to the logical file system thelogical file system contains in meta data of the file and directory structures
If application program doesn't have a required permission of the file then this layer will throw a errorlogical file system also very find path of thefile
Sstf shortest seek time first
Sstf scheduling priority is given to those process which have the shortest even if this request are not thefirst ones in the queue to implement this district

time of every request to calculated in advance in thequeue and then request are scheduling according to their ship time

Scan algorithm:

In scan algorithm head starts from one end to the deals and moves towards the other end search YC servicing request in between one by one and reached file other end then the direction of the head is reversed and the process continues our head continuously SC scan bank for to access this

Conclusion: in this practical I learnt about this scheduling algorithm such as scan s s it observe their working why using in operating system its need for risk management.