Laboratory Manual

For

Applied Operating System

(IT 607)

B.Tech (IT)

SEM VI

June 2010

Faculty of Technology
Dharmsinh Desai University
Nadiad.
www.ddu.ac.in

Table of Contents

Sample experiment

1 AIM: Inter process communication (POSIX-IPC) using shared memory

Problem: Establish Inter process communication using shared memory. Implement day time server and test it using client program.

2 TOOLS/APPARATUS: gcc, libpthread.so library

3 STANDARD PROCEDURES:

3.1 Analyzing the Problem:

Two entities are needed: (1) client (2) server.

Client and server will communicate using shared memory, so, shared memory region is needed.

Shared memory will be used by both server and client, so, for synchronization semaphore is needed.

Server provided time-server facility, so, to access time, time() and ctime() functions are needed.

3.2 Designing the Solution:

1) Server: The steps performed by server entity are as follows.

Get time value

Lock semaphore

Write time value on shared memory

Unlock semaphore

Sleep for 1 ms.

2) Client: The steps performed by client entity are as follows.

Lock the semaphore

Read time value from shared memory

Display time value on console

Unlock the semaphore

3.3 Implementing the Solution

3.3.1 Writing Source Code

//file: shm.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <semaphore.h>
// File permissions
#define FILE_MODE S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH
#define DSIZE 100
//shared structure containing semaphore and date value
struct shared{
        sem_t sem;
        char dateval[DSIZE];
}sharedv;
//main function
int main(int argc,char** argv){
//variable to hold file descriptor of shared memory mapped file
        int fd;
//variable to hold choice entered by user.
        char choice='y';

//variable to hold time value.
        time_t timeval;

//pointer to shared structure to access shared memory region
        struct shared* ptr;

//check that user uses program using proper usage format.
        if(argc!=2){
                fprintf(stderr,"Usage:%s pathname",argv[0]);
                exit(0);
        }

//Open a file
        fd=open(argv[1],O_RDWR|O_CREAT,FILE_MODE);
//Write content of structure to opened file
        write(fd,&sharedv,sizeof(struct shared));

//perform memory mapping of file, and assign returned result to ptr
```

```
        ptr=mmap(NULL,sizeof(struct
shared),PROT_READ|PROT_WRITE,MAP_SHARED,fd,0);

        close(fd);

//initialize semaphore
        sem_init(&ptr->sem,0,1);
        setbuf(stdout,NULL);
        time(&timeval);
        strcpy(ptr->dateval,ctime(&timeval));

//create a new process using fork, assign duty of client to child process
        if(fork()==0){ //child:client
                while(choice=='y'){
//lock semaphore before reading shared memory
                        sem_wait(&ptr->sem);
                        printf("Current date-time is %s\n",ptr->dateval);
//unlock semaphore after reading shared memory
                        sem_post(&ptr->sem);
                        printf("Do you want to know current date, Enter your choice (y/n)=");
                        scanf("%c",&choice);
                        getchar();
                }
                exit(0);
        }
//assign duty of server to parent process
        //parent:server
        while(1){
//get current time value
                time(&timeval);
//lock semaphore before writing to shared memory
                sem_wait(&ptr->sem);
                strcpy(ptr->dateval,ctime(&timeval));
//unlock the semaphore after writing is over
                sem_post(&ptr->sem);
                sleep(1);
        }
}
```

3.3.2 Compilation /Running and Debugging the Solution

1) Compilation:

As we are using system calls related to semaphore operations, we need to use pthread library (/usr/lib/libpthread.so) at the time of compilation. Run following command to compile the program

#gcc shm.c /usr/lib/libpthread.so –o shm

2) Running the program

If we run program by simply writing its name, i.e., ./shm, the program displays the usage

Usage:shm pathname

If we pass name of file (for shared region), the server and client start their activities. Pass following command on command prompt.

./shm date.txt

3.4 Testing the Solution

No input data is needed to test the program. Test the solution by following command.

./shm date.txt

The following output is displayed on console

Fri Mar 15 04:07:28 IST 2002

Do you want to know current date, Enter your choice (y/n)=y

Fri Mar 15 04:07:32 IST 2002

Do you want to know current date, Enter your choice (y/n)=y

Fri Mar 15 04:07:40 IST 2002

4 Conclusions:

Both the client and server perform their activities. As we are using semaphore operations while accessing shared memory, we do not get any corrupted date value as output.

COMMON PROCEDURE

The common procedure for solving programming related problems is as follows:

Step 1: For given problem statement, find out the names of required system calls.
Step 2: Study and understand the usage of those system calls.
Step 3: Design the logic for solving the given problem.
Step 4: Implement the logic in C programming code.
Step 5: Compile the program using make utility
Step 6: Run the program by passing the appropriate command line arguments
Step 7: Note down the output and draw your conclusion.

EXPERIMENT-1

Aim: Study of UNIX commands with all their important options.

Study of following commands

a) Information Management

cal, date, tty, sh, env, set, man, who, whoami

b) Utility commands

wc, echo, tail, less, more, sort, grep, bc, cmp, comm.

c) File System Management

ls, ln, rm, rmdir, mkdir, file, chmod, find, od, pwd, locate, updated, mount, umount, mv

d) Process Management

ps, kill

e) Compilation and debugging

cc, gdb

f) Editors

vi, joe, mcedit, emac

Tools / Apparatus: Linux OS, and command files

Procedure:

1. For each command, read the documentation from man pages, e.g., using man command.
2. Study important options
3. Execute the command with options and study the output.

EXPERIMENT-2

Aim: Program maintenance using make utility.

Tools / Apparatus: Linux OS, make, cc

Procedure:

1. Write a program that is spread over two files.
2. Use following Makefile for program maintenance. To use make utility, use make command.

Makefile

```
test.out: test.o
        cc test.o –o test.out
test.o: test.c test.h
        cc test.c –c
clean:
        rm –f *.o *.*~ *~
run:
        ./test
```

EXPERIMENT-3

Aim: Study system calls related to process & process control
Study system calls: fork(), exec(), wait(), getpid(), getppid(), getuid(), getgid(), geteuid(), getegid()
Problems

1. Print Process ID, Parent process ID, Read user ID, Real group ID, Effective user ID, effective group ID for current process.
2. Write a program that uses fork() and assign some different task to child process. And make sure that parent exits after child finishes its task.
3. Write a program that creates child process and find out/ print values of processID and parent processID for both child and parent processes. Consider following two cases
   (1) Child exits before parent (2) Parent exits before child.

Tools / Apparatus: Linux OS, make, cc

Procedure:
Use common procedure

EXPERIMENT-4

Aim: Study system calls related to file operations.
Study system calls: open(), read(), write(), stat(), fstat()
Problems

1. Implement file copy command using system call also handle all possible errors.
2. Print i-node information using stat/fstat. Also identify type of a file (device file, pipe, directory, link etc.)
3. Find out whether file descriptors and file read pointer are shared or not by parent and child process after fork()

Tools / Apparatus: Linux OS, make, cc

Procedure:
Use common procedure

EXPERIMENT-5

Aim: Study of functions related to threads (POSIX).
Study functions: pthread_create(), pthread_join()
Problem

1. Solve producer consumer problem (bounded buffer) using mutex.

Tools / Apparatus: Linux OS, make, cc, libpthread.so library

Procedure:
Use common procedure

EXPERIMENT-6

Aim: Inter process communication (POSIX-IPC) using pipe
Study system call: pipe()
Problem

1. Client server application using pipes: Client sends file name to server and server sends file content or error as reply to client's request.

Tools / Apparatus: Linux OS, make, cc, libpthread.so library

Procedure:
Use common procedure

EXPERIMENT-7

Aim: Inter process communication (POSIX-IPC) using shared memory
Study system calls: mmap(), shm_open(), shm_unlink()
Problem

1. Establish Inter process communication using shared memory Implement day time server and test it using client program.

Tools / Apparatus: Linux OS, make, cc, libpthread.so library

Procedure:
Use common procedure

EXPERIMENT-8

Aim: Study system calls related to semaphore.
Study system calls: sem_init(), sem_destroy(), sem_wait(), sem_post()
Problem

1. Solve producer consumer problem (multiple producer multiple consumer) using semaphore

Tools / Apparatus: Linux OS, make, cc, libpthread.so library

Procedure:
Use common procedure

EXPERIMENT-9

Aim: Simulation of Process scheduling: Feedback policy.

Tools / Apparatus: Linux OS, make, cc

Procedure:
Use common procedure

EXPERIMENT-10

Aim: Simulation of I/O requests scheduling: Elevator algorithm

Tools / Apparatus: Linux OS, make, cc

Procedure:
Use common procedure

EXPERIMENT-11

Aim: Simulation of deadlock handling: Banker's algorithm

Tools / Apparatus: Linux OS, make, cc

Procedure:
Use common procedure

EXPERIMENT-12

Aim: Simulation of Memory management: LRU page replacement algorithm.

Tools / Apparatus: Linux OS, make, cc

Procedure:
Use common procedure

EXPERIMENT-13

Aim: Installation of Virtual Machine Software and installation of Operating System on Virtual Machine.

Tools / Apparatus: Windows XP OS, Source of Linux OS, Sun Virtual Box

Procedure:

1. Install Sun Virtual Box software on Windows XP OS.
2. Run Virtual Box
3. Click New to set up a new virtual machine profile and follow the wizard.
4. Title your virtual machine. e.g. Red Hat Linux
5. VirtualBox will try to guess how much RAM to allocate for the virtual machine. If you have 1 GB of RAM, 512 MB might be a good allocation.
6. Create a virtual hard drive to install OS.
7. Configure CD ROM settings.
8. Select the newly created virtual machine profile and click Start.
9. After it boots up, click the Install icon on the desktop.
10. Answer all the questions.
11. After installation is over, now you can start/shutdown/use Linux OS on Windows XP OS.

EXPERIMENT-14

Aim: Intercepting and recording the system calls made by a running process using strace tool.

Tools / Apparatus: Linux OS, strace, executable file.

Procedure:

1. Study strace command with its options.
2. Create an executable file or use some existing executable file.
3. Run the executable file using strace. strace -o strace-echo-output.txt echo "Hello there"
4. Note down the output listing of strace generated for selected executable file.
5. Understand the use of system calls by the created process. e.g., echo

Required Reading Materials

Books:

Unix Programming environment By: Kernighan and Pike.

Unix Network programming By: W. Richard Stevens Volume-2

References

- Operating System Concept : Silbertschatz, Galvin, 5ed. ,Addison Wesley.
- Operating system Concepts : Milan Malinkovic, TMH, 2nd ed.
- Operating System : William Stallings, PHI, 2nd ed.