Hands-on List

1. Write a shell script to count the number of block device files in /dev directory.
2. Write a shell program that checks the number of command line arguments and echoes an error message if there are not exactly three arguments or echoes the arguments themselves if there are three.
3. Write a shell program called new_files that will accept a variable number of command line arguments. The shell program will create a new file associated with each command line argument and echo a message that notifies the user as each file is created.
4. Create a directory called .recyclebin in your home directory. Write a shell program called myrm that will move all of the files you delete into the .recyclebin directory, your wastebasket. This is a useful tool which will allow restoration of files after they have been removed. Remember, the UNIX system has no undelete capability.
5. Write a script that uses find to look for a file and echo a suitable message if the file is not found. You must not store the output of the find to a file.
6. Write a script which will give 4 choices to the user 1. ls 2. pwd 3. who 4. exit and execute the command as per the users choice.
7. Write an interactive file-handling shell program that offers the user choice of copying, removing, rename. Once the user has made a choice, the program ask user for the necessary information, such as the file name, new name.

Hands-on List

8. Write shell script that takes a login name as command – line argument and reports when that person logged in.
9. Write a shell script that accepts a file name starting and ending line numbers as arguments and displays all the lines between the given line numbers.
10. Write a script to backup a given directory to a given file name in your home directory. Both, the directory name and the backup file has to be passed as command line input. Design the script with suitable exception handling.
11. Write a script to check how much space is used by each directory of a given file system. The name of the file system has to be provided from the command line parameter.
12. Write a script in /root/myscript.sh according to the following criteria:
    a) If you search for the IIT the output is NIT
    b) If you search for NIT the output is IIT
    c) If you search any other keyword or not give any input, the output is STDERR should be displayed.
13. Write a shell script to print a multiplication table.
14. Write a shell script to print, “Good Morning/Afternoon/Evening based on the current system time.

Hands-on List

15. SED - Write a shell script to perform the following (input file “example” will be given).
    1. For a given file, find all the lines containing our search pattern.
    2. List the lines not containing the search string
    3. Matching lines starting with a given pattern and ending in a second pattern
    4. Print a file starting from a certain line until to the end of file.
    5. Search a given pattern in a file and replace with a new pattern and display the file.
    6. Insert a given string at the beginning of each line of the file.
    7. Insert a given string at the end of each line of the file

16. AWK – Write a Shell script to (The input file “employee.txt” will be given)
    1. Display a given file.
    2. Print the lines which match with a given pattern.
    3. Print only a specific field in the file.
    4. Format a given file with Name, Designation, Department and Salary headings and at the end of a file print Report Generated.
    5. Find the employees, who has id > 200
    6. Find the list of employees in a Technology Department.
    7. Print the number of employees in Technology Department.

HOL 1

Ques 1 to 18 are for File Management

Ques 19 to 30 are for Process Mgmt

CS 513/EC 506 System Software

List of Lab Exercises – Only File and Process Management

---

1. Create the following types of a files using (i) shell command (ii) system call
   a. soft link (symlink system call)
   b. hard link (link system call)
   c. FIFO (mkfifo Library Function or mknod system call)

2. Write a simple program to execute in an infinite loop at the background. Go to /proc directory and identify all the process related information in the corresponding proc directory.

3. Write a program to create a file and print the file descriptor value. Use creat ( ) system call

4. Write a program to open an existing file with read write mode. Try O_EXCL flag also.

5. Write a program to create five new files with infinite loop. Execute the program in the background and check the file descriptor table at /proc/pid/fd.

6. Write a program to take input from STDIN and display on STDOUT. Use only read/write system calls

stdin - 0
stdout - 1

7. Write a program to copy file1 into file2 ($cp file1 file2). using system calls

8. Write a program to open a file in read only mode, read line by line and display each line as it is read. Close the file when end of file is reached.

9. Write a program to print the following information about a given file.

man 2 stat

   a. inode
   b. number of hard links
   c. uid
   d. gid
   e. size
   f. block size
   g. number of blocks
   h. time of last access
   i. time of last modification
   j. time of last change

10. Write a program to open a file with read write mode, write 10 bytes, move the file pointer by 10 bytes (use lseek) and write again 10 bytes.
   a. check the return value of lseek
   b. open the file with od and check the empty spaces in between the data.

11. Write a program to open a file, duplicate the file descriptor and append the file with both the descriptors and check whether the file is updated properly or not.
   a. use dup
   b. use dup2
   c. use fcntl

all questions are done, ticked ques are important

12. Write a program to find out the opening mode of a file. Use fcntl.

13. Write a program to wait for a STDIN for 10 seconds using select. Write a proper print statement to verify whether the data is available within 10 seconds or not (check in $man 2 select).

14 Write a program to find the type of a file.
a. Input should be taken from command line.
b. program should be able to identify any type of a file.

15. Write a program to display the environmental variable of the user (use environ).

16. Write a program to perform mandatory locking.
a. Implement write lock
b. Implement read lock

File Locking Programs!

17. Write a program to simulate online ticket reservation. Implement write lock
Write a program to open a file, store a ticket number and exit. Write a separate program, to open the file, implement write lock, read the ticket number, increment the number and print the new ticket number then close the file.

18. Write a program to perform Record locking.
a. Implement write lock
b. Implement read lock
Create three records in a file. Whenever you access a particular record, first lock it then modify/access to avoid race condition.

Process Management

19. Write a program to find out time taken to execute getpid system call. Use time stamp counter.

20. Find out the priority of your running program. Modify the priority with nice command.

21. Write a program, call fork and print the parent and child process id.

22. Write a program, open a file, call fork, and then write to the file by both the child as well as the parent processes. Check output of the file.

23. Write a program to create a Zombie state of the running program.

24. Write a program to create an orphan process.

25. Write a program to create three child processes. The parent should wait for a particular child (use waitpid system call).

26. Write a program to execute an executable program.
a. use some executable program
b. pass some input to an executable program. (for example execute an executable of $./a.out name)

27. Write a program to execute ls -Rl by the following system calls
a. execl
b. execlp
c. execle

d. execv
e. execvp

28. Write a program to get maximum and minimum real time priority.

29. Write a program to get scheduling policy and modify the scheduling policy (SCHED_FIFO, SCHED_RR).

30. Write a program to run a script at a specific time using a Daemon process.

Quiz 1

EG 301 Operating Systems

Pre-Mid Term Exam

Date: 02 February 2022 **Time: 9:30 to 11am**

Total Marks: 10

- Write a Linux system program to remove duplicate lines for a given file.
- Use system calls and avoid C library functions wherever it is possible.
- Your answer sheet zip file (name_regno.zip) should contain:
  - system program (UniqueLines.c)
  - inputfile (inputfile.txt)
  - screen shot of your program execution output (Refer the given below figure)

```
root@ostl:/home/raju/osassignment# cat inputfile.txt
1. This is a test file for the uniq command.
2. It contains some repeated lines.
3. And some are different.
3. And some are different.
2. It contains some repeated lines.
1. This is a test file for the uniq command.
root@ostl:/home/raju/osassignment#
root@ostl:/home/raju/osassignment# cc UniqueLines.c
root@ostl:/home/raju/osassignment# ./a.out inputfile.txt
1. This is a test file for the uniq command.
2. It contains some repeated lines.
3. And some are different.

Total Lines: 6 & Total Duplicates: 3
root@ostl:/home/raju/osassignment#
```

HOL 2

System Software-II

List of Lab Exercises

1. Write a separate program (for each time domain) to set a interval timer in 10sec and 10micro second
   a. ITIMER_REAL
   b. ITIMER_VIRTUAL
   c. ITIMER_PROF

2. Write a program to print the system resource limits. Use getrlimit system call.

3. Write a program to set (any one) system resource limit. Use setrlimit system call.

4. Write a program to measure how much time is taken to execute 100 getppid ( ) system call. Use time stamp counter.

5. Write a program to print the system limitation of
   a. maximum length of the arguments to the exec family of functions.
   b. maximum number of simultaneous process per user id.
   c. number of clock ticks (jiffy) per second.
   d. maximum number of open files
   e. size of a page
   f. total number of pages in the physical memory
   g. number of currently available pages in the physical memory.

6. Write a simple program to create three threads.

7. Write a simple program to print the created thread ids.

8. Write a separate program using signal system call to catch the following signals.
   a. SIGSEGV
   b. SIGINT
   c. SIGFPE
   d. SIGALRM (use alarm system call)
   e. SIGALRM (use setitimer system call)
   f. SIGVTALRM (use setitimer system call)
   g. SIGPROF (use setitimer system call)

9. Write a program to ignore a SIGINT signal then reset the default action of the SIGINT signal - Use signal system call.

10. Write a separate program using sigaction system call to catch the following signals.
   a. SIGSEGV
   b. SIGINT

c. SIGFPE

11. Write a program to ignore a SIGINT signal then reset the default action of the SIGINT signal - use sigaction system call.

12. Write a program to create an orphan process. Use kill system call to send SIGKILL signal to the parent process from the child process.

13. Write two programs: first program is waiting to catch SIGSTOP signal, the second program will send the signal (using kill system call). Find out whether the first program is able to catch the signal or not.

14. Write a simple program to create a pipe, write to the pipe, read from pipe and display on the monitor.

15. Write a simple program to send some data from parent to the child process.

16. Write a program to send and receive data from parent to child vice versa. Use two way communication.

17. Write a program to execute ls -l | wc.
  a. use dup
  b. use dup2
  c. use fcntl

18. Write a program to find out total number of directories on the pwd.
  execute ls -l | grep ^d | wc ? Use only dup2.

19. Create a FIFO file by
  a. mknod command
  b. mkfifo command
  c. use strace command to find out, which command (mknod or mkfifo) is better.
  c. mknod system call
  d. mkfifo library function

20. Write two programs so that both can communicate by FIFO -Use one way communication.

21. Write two programs so that both can communicate by FIFO -Use two way communications.

22. Write a program to wait for data to be written into FIFO within 10 seconds, use select system call with FIFO.

23. Write a program to print the maximum number of files can be opened within a process and size of a pipe (circular buffer).

24. Write a program to create a message queue and print the key and message queue id.

25. Write a program to print a message queue's (use msqid_ds and ipc_perm structures)
   a. access permission
   b. uid, gid
   c. time of last message sent and received
   d. time of last change in the message queue
   d. size of the queue
   f. number of messages in the queue
   g. maximum number of bytes allowed
   h. pid of the msgsnd and msgrcv

26. Write a program to send messages to the message queue. Check $ipcs -q

27. Write a program to receive messages from the message queue.
   a. with 0 as a flag
   b. with IPC_NOWAIT as a flag

28. Write a program to change the exiting message queue permission. (use msqid_ds structure)

29. Write a program to remove the message queue.

30. Write a program to create a shared memory.
   a. write some data to the shared memory
   b. attach with O_RDONLY and check whether you are able to overwrite.
   c. detach the shared memory
   d. remove the shared memory

31. Write a program to create a semaphore and initialize value to the semaphore.
   a. create a binary semaphore
   b. create a counting semaphore

32. Write a program to implement semaphore to protect any critical section.
   a. rewrite the ticket number creation program using semaphore
   b. protect shared memory from concurrent write access
   c. protect multiple pseudo resources (may be two) using counting semaphore
   d. remove the created semaphore

33. Write a program to communicate between two machines using socket.

34. Write a program to create a concurrent server.
   a. use fork
   b. use pthread_create

Quiz 2

Ref. Hands on List 1, Q 17 and 18

Quiz 2 **EG 301 Operating Systems** **Marks:10**

(Ref: Day1, hands-on-list 17 and 18) Create and Initialize 3 trains data with train name/number and a ticket number. At any point in time, only one process is allowed to book a ticket to a particular train, if some other process tries to book to the same train, it has to wait until the first process exit. The application should allow other processes to book tickets to other trains.

Implement **System V semaphore set** to synchronize the ticket booking.

Note: You should not use file lock (either mandatory or record locking).

Expected Output:

Open 3 terminals and execute the program. The first process books ticket in train number 1 and it is allowed to access the critical section. The second process also tries to book the ticket in train number 1 and it is waiting for the first process exit. At the same time, the third process books ticket in train number 2 and it is allowed to access the critical section.

```
root@raju-VirtualBox:/home/raju/LSP/temp/quiz2# ./a.out
Enter the train number u want to lock>>:  1
semid=1
Before Enter the critical section:
 Inside the critical section:
Waiting for unlocking....Press any key to unlock:
```

```
root@raju-VirtualBox:/home/raju/LSP/temp/quiz2# ./a.out
Enter the train number u want to lock>>:  1
semid=1
Before Enter the critical section:
```

```
root@raju-VirtualBox:/home/raju/LSP/temp/quiz2# ./a.out
Enter the train number u want to lock>>:  2
semid=1
Before Enter the critical section:
 Inside the critical section:
Waiting for unlocking....Press any key to unlock:
```

Include all the source code files (*.txt) and the screenshots in the zip file and upload them into the LMS.

The file name should be <Reg.No_Name>.zip.

EG 301 Operating Systems – Mini Project

Title: *Design and Development of online banking management system*

Description: The project aims to develop a banking system that is user-friendly and multi-functional. The project should have the following functionalities:

a. All account transactional details and customer information are stored in files.
b. Account holders have to pass through a login system to enter their accounts.
c. The application should possess password-protected administrative access; thus preventing the whole management system from unauthorized access.
d. Three types of login should be created: normal user, joint account user; administrator;
e. Once you login as administrator, you should be able to add, delete, modify, search for a specific account details.
f. Once the customer connect to the server, you should get login and password prompt. After successful login, you should get menu for example:
   Do you want to:
   - **Deposit**
   - **Withdraw**
   - **Balance Enquiry**
   - **Password Change**
   - **View details**
   - **Exit**
g. If you login as a joint account holder, proper file locking should be implemented. If you want to view the account details then read lock is to be set else if you want to withdraw or deposit then write lock should be used to protect the critical data section.
h. Use socket programming – Server maintains the data base and service multiple clients concurrently. Client program can connect to the server and access their specific account details.
i. Use system calls instead of Library functions wherever it is possible in the project: Process Management, File Management, File Locking, Multithreading and Inter Process Communication Mechanisms.