OS fundamentals and Data structure training (Version 1.0)

801, Shapath - 1,
Opp. Rajpath Club, S. G. Road,
Ahmedabad – 380054, INDIA,
Phone: +91 - 79 - 40041994
email: info@volansys.com

Contents

History

Version	Author	Date	Comments
1.0	Kavan & HK	25-Jan-2013	First version

Objective

The objective of the training is to learn theory and practical aspect of OS fundamentals and Data structure. At the end of training, trainee should be able to program in multi-threading and multi-process environment, should know about different synchronization and OS concepts. In addition to that trainee should be comfortable with socket programming. Apart from technical aspect, trainee should also learn how to do documentation for software development life cycle.
Training document covers following aspects of OS fundamentals and Data structure.

- Data Structures & Algorithms
  - Abstract data type concept(ADT)
  - Linked List, Stacks, Queues, Trees
  - Sorting, Searching
  - Shortest path
- OS Fundamentals
  - Basic OS Concepts
  - Process
  - Threads (POSIX)
  - Synchronization
  - IPC
  - Sockets

Total Duration

- 28 working days

Total Points

- Total 370 points
- Minimum 270 points required to qualify.

Reference e-book and articles

- Basics Of Linked-List
- Linked List Problems
- Pointers & Memory
- Online book of Data Structure and Algorithms
- Operating Systems Concept 7th Edition A. Silberschatz
- beej_socket_programming

Instructions

- C Training should be completed
- Use C language for development of exercises
- Use Linux OS(Ubuntu 10.04 LTS) for development of all exercises
- Use vim, emacs or similar type of editor
- Use GCC compiler for code compilation with “-Wall” option & “-o” option
- For all process and thread related exercise, use POSIX libraries
- Every exercise must be developed in separate file and its problem-statement should be written at starting of the file

- Answers to each theoretical questions should be given in separate word document
- Every code compilation should not have any error or warning

Evaluation criteria

- All exercises need to be completed within defined duration. If exercises are completed then only evaluation can be done.
- If it is found that the code/design is copied from somewhere, then 50 points will be deducted.
- Proper error handling should be done for every code development exercise.

List of Exercises

Exercise Set#1: Linked-list, Stack, Queue, Sorting, searching

Duration: 7 day
Points Earn: 120

a) Pointer, Memory and Linked-List Theory [2 days,35 points]

Refer: Pointers & Memory, Basics of Linked-List and Linked List Problems

i. List down various use of Pointers. (2 points)
ii. List down all memory allocation techniques available in C language. What are the differences between them? Who has to take care of de-allocating such memories and how? (3 points)
iii. How would you compare array and link list? What are the pros and cons of each one? Which one should be used in which scenario? (5 points)
iv. Write a C program to generate singly link list based on user value and at the end reverse the link-list. (10 points)
v. Write a C program which maintains circular doubly linked list and provides functionality of addition (at beginning, at end, at intermediate), deletion (at beginning, at end, at intermediate) and searching of nodes in this circular doubly linked list. (15 points)

b) Data Structure & Algorithms Theory [1 day,20 points]

Read chapter 1, 2, 3, 4, 6, 7, 8 and 10 of DSA online book and answer following questions:

i. What is reverse polish notation? Where it can be used? (2 points)
ii. What are the preorder, in-order and post-order traversal of binary tree? (3 points)
iii. Why we cannot determine an in-order traversal from given pre-order and post-order traversals? (3 points)
iv. What is the best-suited data structure to convert a prefix notation of an expression to its corresponding post-fix notation and why? (3 points)
v. List the techniques used in handling the collisions in the hash tables. (3 points)
vi. What are the priority queues? How to use them to implement stacks and queues? (3 points)
vii. Explain the shortest path first algorithm. (3 points)

c) Data Structure & Algorithms programming [4 days, 65 points]

i. Write a program for linear search & binary search. Data will be provided by user. Search list can be of random size. For every search display actual timing and memory space information. (10 points)
ii. Write a program for sorting methods: Bubble, Radix and Quick (using Linux library API). Data will be provided by user. Data list can be of random size. For every sorting methods display actual timing and memory space information. (10 points)
iii. Develop a task scheduler module whose job is to maintain tasks as per its priority in a queue. Follow the instruction for more details. (15 points)
   - Minimum priority of task is 0 and maximum priority is 5, lower the priority number means higher the priority of tasks.
   - Task scheduler should dispatch the highest priority task available to end-application on demand.
   - If multiple tasks are at same priority then oldest task available at that priority level should be dispatched first.
   - Scheduler should be such that end-application can add tasks at runtime.
   - Develop a sample end-application to add and demand the task to test the scheduler in multiple ways.

iv. Develop a program to construct a binary search tree (BST). (15 points)

- Tree data will be given by user, user enters EOF to express end of list.
- Display level-order traversal to validate the sequence of integers entered by user.
- Display BST tree in-order, pre-order and post-order traversal format.
- Display timing and memory space information for each type of traversal
- Input the same list in program-i and perform search operation. Compared timing and space information to understand difference between different types of search.

v. Write a program to searches for a line in given directory. (15 points)
- Prepare a hash-table based index for each file of directory.
- Use has table to search specific line provided by user.
- If line matched then display file name along with line number for all occurrences.

Exercise Set#2: OS Concept

Duration: 4 days
Points Earn: 55

a) Basic (What, Why) of OS [1.5 days, 25 points]
Refer Chapter 1 and 2 of OS concept

i. In a multiprogramming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems. What are two such problems? Can we ensure the same degree of security in a time-shared machine as we have in a dedicated machine? Explain your answer. (3 points)
ii. Describe the differences between symmetric and asymmetric multiprocessing. What are three advantages and one disadvantage of multiprocessing systems? (3 points)
iii. What are the differences between a trap and an interrupt? What is the use of each function? (3 points)
iv. When are caches useful? What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device? (5 points)
v. Writing an operating system that can operate without interference from malicious or un-debugged user programs requires some hardware assistance. Name three hardware aids for writing an operating system, and describe how they could be used together to protect the operating system. (3 points)
vi. What is the purpose of a command interpreter? Why is it usually separate from the kernel? (2 points)
vii. List five services provided by an operating system. Explain how each provides convenience to the users. Explain also in which cases it would be impossible for user-level programs to provide these services. (3 points)
viii. What is the main advantage of the layered approach to system design? (3 points)

b) OS Memory & I/O [2.5 days, 30 points]
Refer Chapter 8, 9 and 13 of OS concept

i. Explain the difference between internal and external fragmentation. (2 points)
ii. Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory? (5 points)
iii. On a system with paging, a process cannot access memory that it does not own. Why? How could the operating system allow access to other memory? Why should it or should it not? (5 points)
iv. A certain computer provides its users with a virtual-memory space of 2^{32} bytes. The computer has 2^{18} bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4,096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations. (3 points)
v. Suppose your replacement policy (in a paged system) consists of regularly examining each page and discarding that page if it has not been used since the last examination. What would you gain and what would you lose by using this policy rather than LRU or second-chance replacement? (2 points)
vi. Assume there is an initial 1024 KB segment where memory is allocated using the buddy system. Using Figure 9.27 as a guide, draw the tree illustrating how the following memory requests are allocated. (5 points)
   - Request 240 bytes
   - Request 120 bytes
   - Request 60 bytes
   - Request 130 bytes

Next, modify the tree for the following releases of memory. Perform coalescing whenever possible:

- Release 250 bytes
- Release 60 bytes
- Release 120 bytes

vii. Consider the following I/O scenarios on a single-user PC. (5 points)

- A mouse used with a graphical user interface
- A tape drive on a multitasking operating system (assume no device pre-allocation is available)
- A disk drive containing user files
- A graphics card with direct bus connection, accessible through memory-mapped I/O

For each of these I/O scenarios, would you design the operating system to use buffering, spooling, caching, or a combination? Would you use polled I/O, or interrupt driven I/O? Give reasons for your choices.

viii. UNIX co-ordinates the activities of the kernel I/O components by manipulating shared in-kernel data structures, whereas Windows NT uses object oriented message passing between kernel I/O components. Discuss three pros and three cons of each approach. (3 points)

Exercise Set#3: Process, Thread & Synchronization

Duration: 10 days
Points Earn: 130

a) Process & Thread [1.5 day,15 points]
Read chapter 3 (till section 3.3) & 4 of OS book, "pthread" library document, before start the exercise.

i. Explain your views about Process & Threads. What is advantage/disadvantage of having multi-process over multi-threaded? Suggest one application that would benefit from the use of threads, and one that would not. (3 points)
ii. What are the differences between user-level threads and kernel-supported threads? Under what circumstances is one type "better" than the other? (2 points)
iii. What will happen if parent process does not wait for child process or thread(s) to complete? (2 points)
iv. Developed a parent-child process application, wherein child process prints the number from 1 to 10000 and parent process waits for child process to complete. (3 points)
v. Develop a multi-threaded application to perform 3X3 matrix multiplication, each cell of resultant matrix needs to be calculated in separate thread (5 points)

b) Inter process communication(IPC)[2 days, 40 points]
Read chapter 3 (3.4 onwards) of OS book

i. What is Inter-process Communication (IPC)? Why is it required? What are different IPC mechanisms available in Linux system? (3 points)
ii. Consider the Inter-Process Communication scheme where mailboxes are used. (5 points)
- Suppose a process P wants to wait for two messages, one from mailbox A and one from mailbox B. What sequence of send and receive should it execute?
- What sequence of send and receive should P execute if P wants to wait for one message either from mailbox A or from mailbox B (or from both)?
- A receive operation makes a process wait until the mailbox is nonempty. Either devise a scheme that allows a process to wait until a mailbox is empty, or explain why such a scheme can't exist.

iii. Develop a parent-child process application, wherein child process calculates square of numbers (using progressive arithmetic i.e. square of 2 = 2+2, square of 3 = 3+3+3 and so on) from 1 to 1000 and parent process prints result as soon as soon as child process produce it. Use Pipes for IPC (10 points)
iv. Develop a parent-child process application, wherein child process at random duration calculates square of numbers (via above method) from 1 to 1000 and parent process prints result as soon as soon as child process produce it. Use Shared-memory for IPC (10 points)
v. Develop a simple C based shell, which continuously scans for user input and accordingly execute commands. This shell should validate user inputs. Extra: How would you extend this shell so it provides "|" (Linux pipe) feature? (10+2 points)

c) Synchronization Theory [3 days, 50 points]
Read chapter 6 & 7 of OS book

i. When synchronization between process(s) and/or thread(s) is required? What are synchronization mechanisms available in Linux system? (3 points)

ii. Explain why spinlocks are not appropriate for single-processor systems yet are often used in multiprocessor systems. (2 points)

iii. The Sleeping-Barber Problem. A barbershop consists of a waiting room with n chairs, and the barber room containing the barber chair. If there are no customers to be server, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy, but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a pseudo program to co-ordinate the barber and the customers. (10 points)

iv. Develop a two-threaded application, wherein each thread writes numbers from (1 to 1000) in a same file. Thread one should prefix "#" and thread 2 should prefix "$" before writing numbers. Make sure at a time only one thread can write to a file. Also design this application in such a way that both threads run in parallel. (10 points)

v. Develop a multi threaded producer consumer application. (15 points)
   - Three thread application, wherein each thread invokes at fixed (for 1st thread 10sec, 2nd thread 20sec and 3rd thread 30sec)
   - When thread will invoke, it should read next chunk of 1KB from the target file. After reading chunk apply unique encryption scheme (e.g. add 0x50 on every bytes of 1kb chunk).Each thread having unique encryption scheme. After encryption, dumps encrypted chunk to another file.
   - At a time simultaneously only two threads can read from a target file. If at a time two threads are ready to read then both threads should be able to read data from target file.

vi. Extend above application for status display purpose (10 points)
   - After an iteration of execution thread should notify operation statistics to book-keeping thread.
   - Statistics for each thread are: % read completion, iteration count, number of words read
   - When user generates signal (Ctrl+C or SIGINT), application should terminate gracefully.
   - Hint: Use Condition Variables to notify book-keeping thread, is there any other mechanism to notify book-keeping thread? Which is better and why?

d) Project [3.5 days, 25 points]

Develop a telephonic ticketing system program for XYZ airline for their single 200-seater plane that can:

- Handle callers and distribute them to telephonic ticket agents without allocating two callers to any agent simultaneously. System should have finite number of ticketing agents (more than 3 and less than 7). If no ticket agent is available then caller needs to be put on hold
- Have all tickets agents responding to calls when there are customers waiting to buy tickets
- Assure that only total number of available tickets are sold
- Notify a book-keeper that will perform certain actions once 25%, 50%, 75% and all tickets are sold (and not before)
- It can be assumed that each call will result in successful selling of "one or two or three or four tickets", subject to availability of tickets.
- Doxygen based document is compulsory for this assignment.
- Hint: You can use random-generator function to determine length of calls, no of ticket sold per call

NOTE: Before coding, prepared a requirement and design document for this problem and submit to your mentor. Unless approved by mentor coding should not be started.

Exercise Set#4: Socket Programming

Duration: 7 days
Points Earn: 65

a) Socket Theory [1 day, 10 points]
   i. What is socket? What are different types of socket? (2 points)
   ii. What does IPv4 and IPv6 means? What's the difference between them? (2 points)
   iii. Considering a Connection-less socket based client-server application, what will be a typical sequence of sockets system call for server and client application? (3 points)
   iv. If client-server application is now based on Connection-oriented socket, will there be a change in sequence of system calls? If yes list down sequence of system calls for server and client application. (3 points)

b) Socket Programming[1.5 day, 25 points]
   NOTE:
   - For each exercise select whether you will use connection-oriented or connection-less socket with reason for your choice.
   - By default server should serve multiple-clients at a time

   i. Develop a client server application, wherein client application sends message to server application from STDIN, server echoes' back client Message. Client should display this server response on STDOUT with server processing time. Use special keyword "quit" to close both applications gracefully. NOTE: Message processing time = Message Rx time + Server Processing time + Message Tx Time (10 points)

   ii. Develop a simple file-transfer client server application with following functionalities (20 points):
      - Server on-startup should scan pre-defined directory and create list of files
      - As soon as server can accept client connections it should on demand present the file list to client
      - Server on demand should transfer the selected file to client as soon as possible
      - If demanded, Server should be able to provide file information (meta-data) like size, type, creation date to client.
      - Server should have idle timeout of 120 seconds, i.e. if any client has no activity till specified timeout, server should terminate the connection with that client
      - Client on start-up or as demanded by user should fetch file list from server
      - As per file selected by user, client should ask server to transfer that file
      - If demanded by user, client should fetch information of file from server

c) chat application [4.5 day, 30 points)
   Develop a console based chat application wherein users can chat simultaneously with each other inside intranet.

   NOTE: Before implementing develop a requirement and design document for this problem and submit it to your mentor. Unless approved by mentor implementation should not be started. Doxygen based document is compulsory for this assignment.