Started on	Saturday, 22 May 2021, 8:00 AM
State	Finished
Completed on	Saturday, 22 May 2021, 9:30 AM
Time taken	1 hour 30 mins
Grade	**26.12** out of 40.00 (**65**%)

Question **1**

Incorrect

Mark 0.00 out of 1.00

A 4 GB disk with 1 KB of block size would require these many number of **blocks** for it's free block bitmap:

Answer: 4096 ✖

The correct answer is: 512

Question **2**

Correct

Mark 1.00 out of 1.00

Given that the memory access time is 110 ns, probability of a page fault is 0.5 and page fault handling time is 12 ms,
The effective memory access time in nanoseconds is:

Answer: 6000165 ✔

The correct answer is: 6000055.00

Question **3**

Incorrect

Mark 0.00 out of 1.00

The maximum size of a file in number of blocks of BSIZE in xv6 code is

(write a number only)

Answer: 268 ✖

The correct answer is: 138

Question **4**

Incorrect

Mark 0.00 out of 1.00

Calculate the average waiting time using
Round Robin scheduling with time quantum of 5 time units
for the following workload

assuming that they arrive in the order written below.

Process Burst Time
P1 5
P2 7
P3 6
P4 2

Write only a number in the answer upto two decimal points.

Answer: 40.75 ✖

The correct answer is: 10.25

Question **5**

Correct

Mark 1.00 out of 1.00

For the reference string

4 2 5 1 0 1 2 5 4 1 2

the number of page faults, including initial ones,

with FIFO replacement and 2 frames are :

Answer: 10 ✔

4 -

4 2

5 2

5 1

0 1

-

2 1

2 5

4 5

4 1

2 1

The correct answer is: 10

Question **6**

Correct

Mark 1.00 out of 1.00

Assuming a 16- KB page size, what is the page number for the address 428517 reference in decimal :

(give answer also in decimal)

Answer: 27 ✔

The correct answer is: 26

Question **7**

Correct

Mark 1.00 out of 1.00

In the code below assume that each function can be executed concurrently by many threads/processes.
Ignore syntactical issues, and focus on the semantics.

This program is an example of

```
spinlock a, b;  // assume initialized
thread1() {
    spinlock(b);
    //some code;
    spinlock(a);
    //some code;
    spinunlock(b);
    spinunlock(a);
}
thread2() {
    spinlock(a);
    //some code;
    spinlock(b);
    //some code;
    spinunlock(b);
    spinunlock(a);
}
```

- a. Deadlock ✔
- b. Self Deadlock
- c. None of these
- d. Deadlock or livelock depending on actual race
- e. Livelock

Your answer is correct.

The correct answer is: Deadlock

Question **8**

Partially correct

Mark 1.33 out of 2.00

Match the snippets of xv6 code with the core functionality they achieve, or problems they avoid.
"..." means some code.

```
static inline uint
xchg(volatile uint *addr, uint newval)
{
  uint result;

  // The + in "+m" denotes a read-modify-write operand.
  asm volatile("lock; xchgl %0, %1" :
               "+m" (*addr), "=a" (result) :
               "1" (newval) :
               "cc");
  return result;
}
```

Atomic compare and swap instruction (to be expanded inline into code) ✔

```
void
sleep(void *chan, struct spinlock *lk)
{
  ...
  if(lk != &ptable.lock){
    acquire(&ptable.lock);
    release(lk);
  }
```

If you don't do this, a process may be running on two processors parallely ✖

```
void
acquire(struct spinlock *lk)
{
...
__sync_synchronize();
```

Tell compiler not to reorder memory access beyond this line ✔

Your answer is partially correct.

You have correctly selected 2.

The correct answer is: static inline uint
xchg(volatile uint *addr, uint newval)
{
  uint result;

  // The + in "+m" denotes a read-modify-write operand.
  asm volatile("lock; xchgl %0, %1" :
               "+m" (*addr), "=a" (result) :
               "1" (newval) :
               "cc");
  return result;
} → Atomic compare and swap instruction (to be expanded inline into code), void
sleep(void *chan, struct spinlock *lk)
{
  ...
  if(lk != &ptable.lock){
    acquire(&ptable.lock);
    release(lk);
  } → Avoid a self-deadlock, void
acquire(struct spinlock *lk)
{
...
__sync_synchronize(); → Tell compiler not to reorder memory access beyond this line

Question **9**

Correct

Mark 1.00 out of 1.00

Predict the output of the program given here.

Assume that all the path names for the programs are correct. For example "/usr/bin/echo" will actually run echo command.

Assume that there is no mixing of printf output on screen if two of them run concurrently.

In the answer replace a new line by a single space.

For example::

`good`

`output`

should be written as `good output`

`--`

```
main() {
    int i;
    i = fork();
    if(i == 0)
        execl("/usr/bin/echo", "/usr/bin/echo", "hi", 0);
    else
        wait(0);
    fork();
    execl("/usr/bin/echo", "/usr/bin/echo", "one", 0);
}
```

Answer: hi one one ✔

The correct answer is: hi one one

Question **10**

Partially correct

Mark 1.67 out of 2.00

Select all the blocks that may need to be written back to disk (if updated, of-course), as "Yes", when an operation of deleting a file is carried out on ext2 file system.

An option has to be correct entirely to be marked "Yes"

Superblock	Yes ✔
One or multiple data blocks of the parent directory	No ✔
One or more data bitmap blocks for the parent directory	No ✔
Block bitmap(s) for all the blocks of the file	No ✖
Possibly one block bitmap corresponding to the parent directory	Yes ✔
Data blocks of the file	No ✔

Your answer is partially correct.

only one data block of parent directory. multiple blocks not possible. an entry is always contained within one single block

You have correctly selected 5.

The correct answer is: Superblock → Yes, One or multiple data blocks of the parent directory → No, One or more data bitmap blocks for the parent directory → No, Block bitmap(s) for all the blocks of the file → Yes, Possibly one block bitmap corresponding to the parent directory → Yes, Data blocks of the file → No

Question **11**

Correct

Mark 1.00 out of 1.00

Select all the correct statements about bootloader.

Every wrong selection will deduct marks proportional to 1/n where n is total wrong choices in the question.

You will get minimum a zero.

- [x] a. Modern Bootloaders often allow configuring the way an OS boots ✔
- [x] b. Bootloaders allow selection of OS to boot from ✔
- [ ] c. Bootloader must be one sector in length
- [ ] d. The bootloader loads the BIOS
- [x] e. LILO is a bootloader ✔

Your answer is correct.

The correct answers are: LILO is a bootloader, Modern Bootloaders often allow configuring the way an OS boots, Bootloaders allow selection of OS to boot from

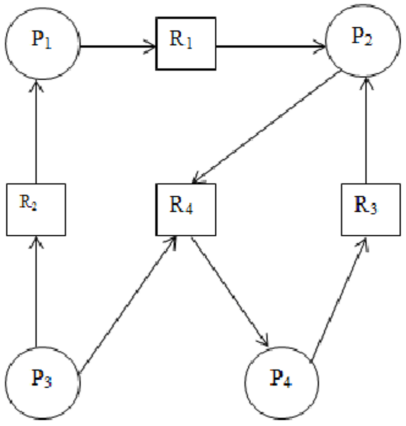
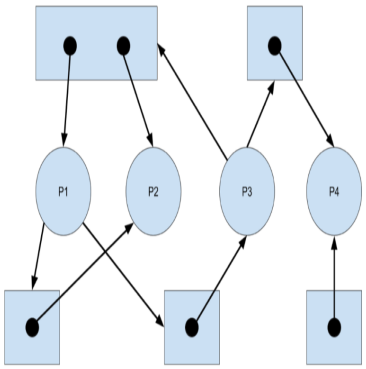
Question **12**

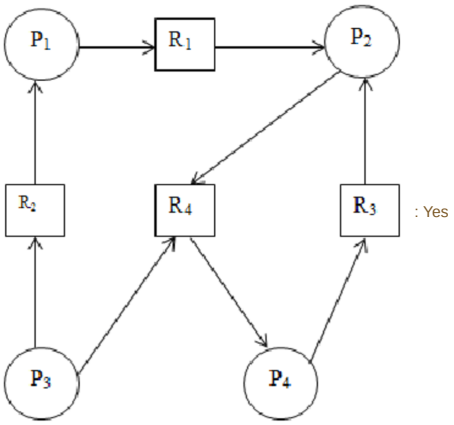
Incorrect

Mark 0.00 out of 1.00

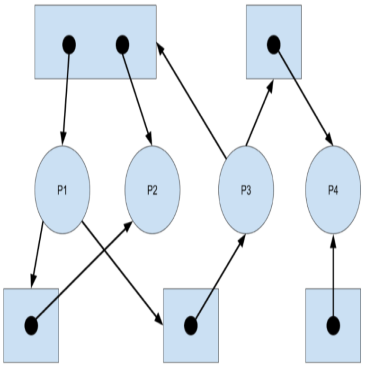
For each of the resource allocation diagram shown,

infer whether the graph contains at least one deadlock or not.

Yes	No	
		
		



: Yes



: No

Question **13**

Partially correct

Mark 0.71 out of 1.00

Mark the statements about device drivers by marking as True or False.

True	False		
○	●	It's possible that a particular hardware has multiple device drivers available for it.	✖
●	○	xv6 has device drivers for IDE disk and console.	✔
●	○	A disk driver converts OS's logical view of disk into physical locations on disk.	✔
●	○	A device driver code is specific to a hardware device	✔
●	○	All devices of the same type (e.g. 2 hard disks) can typically use the same device driver	✔
○	●	Writing a device driver mandatorily demands reading the technical documentation about the hardware.	✖
○	●	Device driver is an intermediary between the end-user and OS	✔

It's possible that a particular hardware has multiple device drivers available for it.: True
xv6 has device drivers for IDE disk and console.: True
A disk driver converts OS's logical view of disk into physical locations on disk.: True
A device driver code is specific to a hardware device: True
All devices of the same type (e.g. 2 hard disks) can typically use the same device driver: True
Writing a device driver mandatorily demands reading the technical documentation about the hardware.: True
Device driver is an intermediary between the end-user and OS: False

Question **14**

Partially correct

Mark 0.33 out of 1.00

Consider this program.
Some statements are identified using the // comment at the end.

Assume that = is an atomic operation.

```
#include <stdio.h>
#include <pthread.h>
long c = 0, c1 = 0, c2 = 0, run = 1;
void *thread1(void *arg) {
    while(run == 1) {//E
        c = 10; //A
        c1 = c2 + 5; //B
    }
}
void *thread2(void *arg) {
    while(run == 1) {//F
        c = 20;//C
        c2 = c1 + 3;//D
    }
}
int main() {
    pthread_t th1, th2;
    pthread_create(&th1, NULL, thread1, NULL);
    pthread_create(&th2, NULL, thread2, NULL);
    sleep(2);
    run = 0;
    fprintf(stdout, "c = %ld c1+c2 = %ld c1 = %ld c2 = %ld \n", c, c1+c2, c1, c2);
    fflush(stdout);
}
```

Which statements are part of the critical Section?

Yes	No		
○	✓	F	✗
✓	○	D	✓
○	✓	C	✗
○	✓	A	✗
✓	○	B	✓
○	✓	E	✗

F: No
D: Yes
C: No
A: No
B: Yes
E: No

Question **15**

Partially correct

Mark 1.43 out of 2.00

Mark statements as T/F

All statements are in the context of preventing deadlocks.

True	False		
◉	○	A process holding one resources and waiting for just one more resource can also be involved in a deadlock.	✔
◉	○	If a resource allocation graph contains a cycle then there is a guarantee of a deadlock	✖
◉	○	The lock ordering to be followed to avoid circular wait is a code in OS that checks for compliance with decided order	✖
◉	○	Circular wait is avoided by enforcing a lock ordering	✔
◉	○	Hold and wait means a thread/process holding some locks and waiting for acquiring some.	✔
◉	○	Deadlock is possible if all the conditions are met at the same time: Mutual exclusion, hold and wait, no pre-emption, circular wait.	✔
◉	○	Mutual exclusion is a necessary condition for deadlock because it brings in locks on which deadlock happens	✔

A process holding one resources and waiting for just one more resource can also be involved in a deadlock.: True
If a resource allocation graph contains a cycle then there is a guarantee of a deadlock: False
The lock ordering to be followed to avoid circular wait is a code in OS that checks for compliance with decided order: False
Circular wait is avoided by enforcing a lock ordering: True
Hold and wait means a thread/process holding some locks and waiting for acquiring some.: True
Deadlock is possible if all the conditions are met at the same time: Mutual exclusion, hold and wait, no pre-emption, circular wait.: True
Mutual exclusion is a necessary condition for deadlock because it brings in locks on which deadlock happens: True

Question **16**

Correct

Mark 1.00 out of 1.00

Match the left side use(or non-use) of a synchronization primitive with the best option on the right side.

This is the smallest primitive made available in software, using the hardware provided atomic instructions	spinlock ✔
This tool is useful for event-wait scenarios	semaphore ✔
This tool is more useful on multiprocessor systems	spinlock ✔
This tool is quite attractive in solving the main bounded buffer problem	semaphore ✔
This tool is very useful for waiting for 'something'	condition variables ✔

Your answer is correct.

The correct answer is: This is the smallest primitive made available in software, using the hardware provided atomic instructions → spinlock, This tool is useful for event-wait scenarios → semaphore, This tool is more useful on multiprocessor systems → spinlock, This tool is quite attractive in solving the main bounded buffer problem → semaphore, This tool is very useful for waiting for 'something' → condition variables

Question **17**

Correct

Mark 1.00 out of 1.00

The permissions -rwx--x--x on a file mean

- [x] a. The file can be read only by the owner ✔
- [ ] b. 'cat' on the file by owner will not work
- [ ] c. 'cat' on the file by any user will work
- [x] d. 'rm' on the file by any user will work ✔
- [x] e. The file can be executed by anyone ✔
- [x] f. The file can be written only by the owner ✔

Your answer is correct.

The correct answers are: The file can be executed by anyone, The file can be read only by the owner, The file can be written only by the owner, 'rm' on the file by any user will work

Question **18**

Incorrect

Mark 0.00 out of 1.00

Note: for this question you get full marks if you select all and only correct options, you get ZERO if at least one option is wrong or not selected.

Select all the correct statements about log structured file systems.

- [x] a. a transaction is said to be committed when all operations are written to file system ✖
- [x] b. log may be kept on same block device or another block device ✔
- [ ] c. file system recovery may end up losing data
- [x] d. even if file systems followed immediate writes (i.e. non-delayed writes), it could still require recovery ✔
- [x] e. file system recovery recovers all the lost data ✖

Your answer is incorrect.

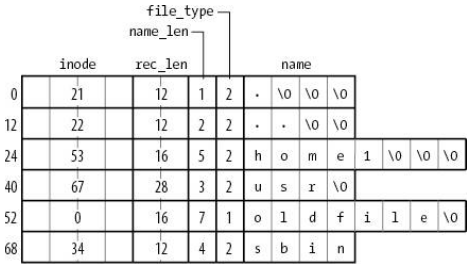
The correct answers are: file system recovery may end up losing data, log may be kept on same block device or another block device, even if file systems followed immediate writes (i.e. non-delayed writes), it could still require recovery

Question **19**

Incorrect

Mark 0.00 out of 1.00

Consider the structure of directory entry in ext2, as shown in this diagram.



Select the correct statements about the directory entry in ext2 file system.

The correct formula for rec_len is (when entries are continuously stored)

- a. rec_len = sizeof(inode entry) + sizeof(name len entry) + sizeof(file type entry) + (strlen(name) + (-1) * (strlen(name)%4)
- b. rec_len = sizeof(inode entry) + sizeof(name len entry) + sizeof(file type entry) + (strlen(name) + (strlen(name)- 4) %4
- c. rec_len = sizeof(inode entry) + sizeof(name len entry) + sizeof(file type entry) + (strlen(name) + 4 - (strlen(name)%4) ✖
- d. rec_len = sizeof(inode entry) + sizeof(name len entry) + sizeof(file type entry) + (strlen(name) + (-1) * (strlen(name)- 4)
- e. rec_len = sizeof(inode entry) + sizeof(name len entry) + sizeof(file type entry) + (strlen(name) %4
- f. rec_len = sizeof(inode entry) + sizeof(name len entry) + sizeof(file type entry) + strlen(name)

Your answer is incorrect.

The correct answer is: rec_len = sizeof(inode entry) + sizeof(name len entry) + sizeof(file type entry) + (strlen(name) + (-1) * (strlen(name)- 4)

Question **20**

Partially correct

Mark 0.50 out of 1.00

Mark whether the given sequence of events is possible or not-possible. Also, select the reason for your answer.

For each sequence it's a not-possible sequence if some important event is not mentioned in the sequence.

Assume that the kernel code is non-interruptible and uniprocessor system.

```
Process P1 executing a system call
Timer interrupt
Generic interrupt handler runs
Scheduler runs
Scheduler selects P2 for execution
P2 returns from timer interrupt handler
Process p2, user code executing
```

This sequence of events is: not-possible ✔

Because

Interrupts are not possible when kernel code is running ✖

Question **21**

Incorrect

Mark 0.00 out of 1.00

The given semaphore implementation faces which problem?

Assume any suitable code for signal()

Note: blocks means waits in a wait queue.

```
struct semaphore {
        int val;
        spinlock lk;
};
sem_init(semaphore *s, int initval) {
        s->val = initval;
        s->sl = 0;
}
wait(semaphore *s) {
        spinlock(&(s->sl));
        while(s->val <=0)
        ;
        (s->val)--;
        spinunlock(&(s->sl));
}
```

a. blocks holding a spinlock

b. deadlock

c. too much spinning, bounded wait not guaranteed ✖

d. not holding lock after unblock

Your answer is incorrect.

The correct answer is: deadlock

Question **22**

Partially correct

Mark 0.80 out of 1.00

Mark statements True/False w.r.t. change of states of a process.

Reference: The process state diagram (and your understanding of how kernel code works)

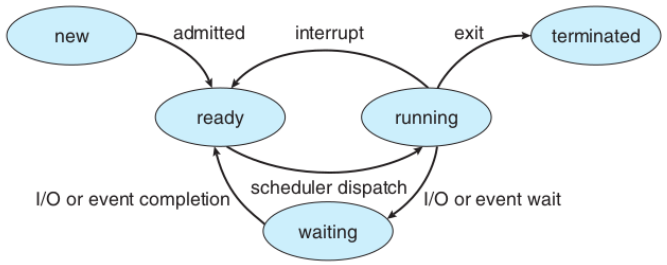


Figure 3.2 Diagram of process state.

True	False		
○	●	A process in RUNNING state only can become TERMINATED because scheduler moves it to ZOMBIE state	✔
●	○	A process in READY state can not go to WAITING state because the resource on which it will WAIT will not be in use when process is in READY state.	✖
●	○	A process in WAITING state can not become RUNNING because the event it's waiting for has not occurred	✔
●	○	Every process has to go through ZOMBIE state, at least for a small duration.	✔
●	○	Only a process in READY state is considered by scheduler	✔

A process in RUNNING state only can become TERMINATED because scheduler moves it to ZOMBIE state: False
A process in READY state can not go to WAITING state because the resource on which it will WAIT will not be in use when process is in READY state.: False
A process in WAITING state can not become RUNNING because the event it's waiting for has not occurred: True
Every process has to go through ZOMBIE state, at least for a small duration.: True
Only a process in READY state is considered by scheduler: True

Question **23**

Correct

Mark 1.00 out of 1.00

Select T/F for statements about Volume Managers.
Do pay attention to the use of the words physical partition and physical volume.

True	False			
◉	○	The volume manager can create further internal sub-divisions of a physical partition for efficiency or features.	✔	
◉	○	A logical volume can be extended in size but upto the size of volume group	✔	
◉	○	A logical volume may span across multiple physical volumes	✔	
◉	○	The volume manager stores additional metadata on the physical disk partitions	✔	
◉	○	A physical partition should be initialized as a physial volume, before it can be used by volume manager.	✔	
◉	○	A volume group consists of multiple physical volumes	✔	
◉	○	A logical volume may span across multiple physical partitions	✔	since a physical volume is made up of physical partitions, and a volume can span across multiple PVs, it can also span across multiple PP

The volume manager can create further internal sub-divisions of a physical partition for efficiency or features.: True
A logical volume can be extended in size but upto the size of volume group: True
A logical volume may span across multiple physical volumes: True
The volume manager stores additional metadata on the physical disk partitions: True
A physical partition should be initialized as a physial volume, before it can be used by volume manager.: True
A volume group consists of multiple physical volumes: True
A logical volume may span across multiple physical partitions: True

Question **24**

Correct

Mark 1.00 out of 1.00

Map the block allocation scheme with the problem it suffers from

(Match pairs 1-1, match a scheme with the problem that it suffers from relatively the most, compared to others)

Continuous allocation	need for compaction	✔
Linked allocation	Too many seeks	✔
Indexed Allocation	Overhead of reading metadata blocks	✔

Your answer is correct.

The correct answer is: Continuous allocation → need for compaction, Linked allocation → Too many seeks, Indexed Allocation → Overhead of reading metadata blocks

Question **25**

Correct

Mark 1.00 out of 1.00

This one is not a system call:

- ○ a. open
- ○ b. read
- ○ c. write
- ◉ d. scheduler ✔

Your answer is correct.

The correct answer is: scheduler

Question **26**

Correct

Mark 1.00 out of 1.00

Match the pairs.

This question is based on your general knowledge about operating systems/related concepts and their features.

Java thereads	monitors,re-entrant locks, semaphores ✔
Linux threads	atomic-instructions, spinlocks, etc. ✔
POSIX threads	semaphore, mutex, condition variables ✔

Your answer is correct.

The correct answer is: Java thereads → monitors,re-entrant locks, semaphores, Linux threads → atomic-instructions, spinlocks, etc., POSIX threads → semaphore, mutex, condition variables

Question **27**

Correct

Mark 1.00 out of 1.00

Consider the following list of free chunks, in continuous memory management:

7k, 15k, 21k, 14k, 19k, 6k

Suppose there is a request for chunk of size 5k, then the free chunk selected under each of the following schemes will be

Best fit: 6k ✔

First fit: 7k ✔

Worst fit: 21k ✔

Question **28**

Correct

Mark 1.00 out of 1.00

This one is not a scheduling algorithm

- a. Round Robin
- b. SJF
- c. Mergesort ✔
- d. FCFS

Your answer is correct.

The correct answer is: Mergesort

Question **29**

Correct

Mark 1.00 out of 1.00

Mark whether the concept is related to scheduling or not.

Yes	No		
◉	○	timer interrupt	✔
◉	○	context-switch	✔
◉	○	ready-queue	✔
○	◉	file-table	✔
◉	○	runnable process	✔

timer interrupt: Yes
context-switch: Yes
ready-queue: Yes
file-table: No
runnable process: Yes

Question **30**

Partially correct

Mark 1.00 out of 2.00

Map ext2 data structure features with their purpose

Many copies of Superblock	Choose...
Free blocks count in superblock and group descriptor ✖	Redundancy to ensure the most crucial data structure is not lost
Used directories count in group descriptor ✖	is redundant and helps do calculations of directory entries faster
Combining file type and access rights in one variable ✔	saves 1 byte of space
rec_len field in directory entry ✖	Try to keep all the data of a directory and it's file close together in a group
File Name is padded ✔	aligns all memory accesses on word boundary, improving performance
Inode bitmap is one block ✔	limits total number of files that can belong to a group
Block bitmap is one block ✔	Limits the size of a block group, thus improvising on purpose of a group
Mount count in superblock ✔	to enforce file check after certain amount of mounts at boot time
Inode table location in Group Descriptor ✖	is redundant and helps do calculations of directory entries faster
Inode table ✔	All inodes are kept together so that one disk read leads to reading many inodes together, effectively doing a buffering of subsequent inode reads, and to save space on disk
A group ✖	Redundancy to ensure the most crucial data structure is not lost

Your answer is partially correct.

You have correctly selected 6.

The correct answer is: Many copies of Superblock → Redundancy to ensure the most crucial data structure is not lost, Free blocks count in superblock and group descriptor → Redundancy to help fsck restore consistency, Used directories count in group descriptor → attempt is made to evenly spread the first-level directories, this count is used there, Combining file type and access rights in one variable → saves 1 byte of space, rec_len field in directory entry → allows holes and linking of entries in directory, File Name is padded → aligns all memory accesses on word boundary, improving performance, Inode bitmap is one block → limits total number of files that can belong to a group, Block bitmap is one block → Limits the size of a block group, thus improvising on purpose of a group, Mount count in superblock → to enforce file check after certain amount of mounts at boot time, Inode table location in Group Descriptor → Obvious, as it's per group and not per file-system, Inode table → All inodes are kept together so that one disk read leads to reading many inodes together, effectively doing a buffering of subsequent inode reads, and to save space on disk, A group → Try to keep all the data of a directory and it's file close together in a group

Question **31**

Partially correct

Mark 1.85 out of 2.00

Mark True/False

Statements about scheduling and scheduling algorithms

True	False		
◉	○	The nice() system call is used to set priorities for processes	✔
◉	○	Aging is used to ensure that low-priority processes do not starve in priority scheduling.	✔
○	◉	In non-pre-emptive priority scheduling, the highest priority process is scheduled and runs until it gives up CPU.	✖
◉	○	xv6 code does not care about Processor Affinity	✔
◉	○	In pre-emptive priority scheduling, priority is implemented by assigning more time quantum to higher priority process.	✔
◉	○	A scheduling algorithm is non-premptive if it does context switch only if a process voluntarily relinquishes CPU or it terminates.	✔
◉	○	Processor Affinity refers to memory accesses of a process being stored on cache of that processor	✔
◉	○	Response time will be quite poor on non-interruptible kernels	✔
◉	○	Shortest Remaining Time First algorithm is nothing but pre-emptive Shortest Job First algorithm	✔
◉	○	On Linuxes the CPU utilisation is measured as the time spent in scheduling the idle thread	✔
◉	○	Generally the voluntary context switches are much more than non-voluntary context switches on a Linux system.	✔
◉	○	Pre-emptive scheduling leads to many race conditions in kernel code.	✔
◉	○	Statistical observations tell us that most processes have large number of small CPU bursts and relatively smaller numbers of large CPU bursts.	✔

The nice() system call is used to set priorities for processes: True
Aging is used to ensure that low-priority processes do not starve in priority scheduling.: True
In non-pre-emptive priority scheduling, the highest priority process is scheduled and runs until it gives up CPU.: True
xv6 code does not care about Processor Affinity: True
In pre-emptive priority scheduling, priority is implemented by assigning more time quantum to higher priority process.: True
A scheduling algorithm is non-premptive if it does context switch only if a process voluntarily relinquishes CPU or it terminates.: True
Processor Affinity refers to memory accesses of a process being stored on cache of that processor: True
Response time will be quite poor on non-interruptible kernels: True
Shortest Remaining Time First algorithm is nothing but pre-emptive Shortest Job First algorithm: True
On Linuxes the CPU utilisation is measured as the time spent in scheduling the idle thread: True
Generally the voluntary context switches are much more than non-voluntary context switches on a Linux system.: True
Pre-emptive scheduling leads to many race conditions in kernel code.: True
Statistical observations tell us that most processes have large number of small CPU bursts and relatively smaller numbers of large CPU bursts.: True

Question **32**

Partially correct

Mark 1.17 out of 2.00

The unix file semantics demand that changes to any open file are visible immediately to any other processes accessing that file at that point in time.

Select the data-structure/programmatic features that ensure the implementation of unix semantics. (Assume there is no mmap())

Yes	No		
○ ✗	◉ ☑	All processes accessing the same file share the file descriptor among themselves	✔
○ ✗	◉ ☑	The pointer entry in the file descriptor array entry points to the data of the file directly	✔
◉ ✗	○ ☑	There is only one global file structure per on-disk file.	✖
○ ✗	◉ ☑	All file accesses are made using only global variables	✔
◉ ✗	○ ☑	The 'file offset' is shared among all the processes that access the file.	✖
○ ✗	◉ ☑	No synchronization is implemented so that changes are made available immediately.	✔
◉ ✗	○ ☑	A single spinlock is to be used to protect the unique global 'file structure' representing the file, thus synchronizing access, and making other processes wait for earlier process to finish writing so that writes get visible immediately.	✖
◉ ☑	○ ✗	There is only one in-memory copy of the on disk file's contents in kernel memory/buffers	✔
◉ ✗	○ ☑	The file descriptors in every PCB are pointers to the same global file structure.	✖
○ ✗	◉ ☑	The file descriptor array is external to PCB and all processes that share a file, have pointers to same file-descriptors' array	✔
◉ ☑	○ ✗	All file structures representing any open file, give access to the same in-memory copy of the file's contents	✔
◉ ✗	○ ☑	The 'file offset' index is stored outside the file-structure to which file-descriptor array points	✖

All processes accessing the same file share the file descriptor among themselves: No
The pointer entry in the file descriptor array entry points to the data of the file directly: No
There is only one global file structure per on-disk file.: No
All file accesses are made using only global variables: No
The 'file offset' is shared among all the processes that access the file.: No
No synchronization is implemented so that changes are made available immediately.: No
A single spinlock is to be used to protect the unique global 'file structure' representing the file, thus synchronizing access, and making other processes wait for earlier process to finish writing so that writes get visible immediately.: No
There is only one in-memory copy of the on disk file's contents in kernel memory/buffers: Yes
The file descriptors in every PCB are pointers to the same global file structure.: No
The file descriptor array is external to PCB and all processes that share a file, have pointers to same file-descriptors' array: No
All file structures representing any open file, give access to the same in-memory copy of the file's contents: Yes
The 'file offset' index is stored outside the file-structure to which file-descriptor array points: No

Question **33**

Partially correct

Mark 0.33 out of 2.00

Map the function in xv6's file system code, to it's perceived logical layer.

namei	inode	✗
filestat()	Choose...	
dirlookup	directory	✓
ialloc	file descriptor	✗
stati	Choose...	
ideintr	buffer cache	✗
bread	Choose...	
balloc	file descriptor	✗
sys_chdir()	system call	✓
skipelem	system call	✗
commit	system call	✗
bmap	system call	✗

Your answer is partially correct.

You have correctly selected 2.
The correct answer is: namei → pathname lookup, filestat() → file descriptor, dirlookup → directory, ialloc → inode, stati → inode, ideintr → disk driver, bread → buffer cache, balloc → block allocation on disk, sys_chdir() → system call, skipelem → pathname lookup, commit → logging, bmap → inode

◄ Course Exit Feedback

Jump to...

xv6-public-master ►

Kd-trees

Amit Shesh
Northeastern University

Build a K-d tree from points

Algorithm

Function *BuildTree*($P[1 \dots n]$)**:**

- $X \leftarrow$ array of indices that sort P by x-coordinate
- $Y \leftarrow$ array of indices that sort P by y-coordinate
- **Return** BuildKDTree_rec($P, X, Y, 0, 1$)

Build a K-d tree from points

Algorithm

Function *BuildTree_rec*$(P, X, Y, depth, numPointsPerLeaf)$**:**

- **If** $size(X) \leq numPointsPerLeaf$ **:**
  - **Return** new Leaf(P, X)
- **If** *depth is even* **:**
  - `// Split with vertical line`
  1. $l \leftarrow x - c = 0$ where c is the median x-coordinate
  2. $X_{left} \leftarrow$ all entries in X for points to left of l
  3. $Y_{left} \leftarrow$ all entries in Y for points to left of l
  4. $X_{right} \leftarrow$ all entries in X for points to right of l
  5. $Y_{right} \leftarrow$ all entries in Y for points to right of l
  6. $on \leftarrow$ all entries in X (or Y) for points on l
  7. $left \leftarrow$ BuildTree_rec$(P, X_{left}, Y_{left}, depth + 1, numPointsPerLeaf)$
  8. $right \leftarrow$ BuildTree_rec$(P, X_{right}, Y_{right}, depth + 1, numPointsPerLeaf)$
  9. **Return** new Node$(on, left, right, l)$
- **Else**
  - `// mirror case`
  10. $l \leftarrow y - c = 0$ where c is the median y-coordinate
  11. $X_{left} \leftarrow$ all entries in X for points below l
  12. $Y_{left} \leftarrow$ all entries in Y for points below l
  13. $X_{right} \leftarrow$ all entries in X for points above l
  14. $Y_{right} \leftarrow$ all entries in Y for points above l
  15. $on \leftarrow$ all entries in X (or Y) for points on l
  16. $left \leftarrow$ BuildTree_rec$(P, X_{left}, Y_{left}, depth + 1, numPointsPerLeaf)$
  17. $right \leftarrow$ BuildTree_rec$(P, X_{right}, Y_{right}, depth + 1, numPointsPerLeaf)$
  18. **Return** new Node$(on, left, right, l)$

Range Query

Algorithm

```
Function PointsInRange(root, C, r):
    If root is a leaf :
        S ← []
        For each point P in root
            If dist(P, C) ≤ r :
                Add P to S
        Return S
    Else
        // find the signed distance of C from split
           line
19      sd ← a ∗ C.x + b ∗ C.y + c
20      If sd < 0 :
21          main ← left(root)
22          other ← right(root)
23      Else
24          main ← right(root)
25          other ← left(root)
26      S ← []
        // First find all points in main that are in
           range
27      S ← PointsInRange(main, C, r)
        // If range crosses the split line, look on
           other side as well
28      If |sd| < r :
29          S ← S+ all points in root closer to C than r
30          S ← S+ PointsInRange(other, C, r)
31      Return S
```

Nearest point query

Algorithm

```
Function NearestPoint(root, Q):
    If root is a leaf :
        n ← points[1] For each point p in root
            If p is closer to Q than n :
                n ← p
        Return p
    Else
        // find the signed distance of C from split
           line
32      sd ← a ∗ C.x + b ∗ C.y + c
33      If sd < 0 :
34          main ← left(root)
35          other ← right(root)
36      Else
37          main ← right(root)
38          other ← left(root)
        // Find nearest point in main
39      n ← NearestPoint(main, Q)
40      If n is nil :
            // If no point was found in main, start
               with point on split line
41          n ← a point in root
        // Consider a circle centered at Q passing
           through n
        // If this circle crosses split line, search
           in other child
42      If |sd| ≤ dist(n, Q) :
43          Update n to be itself or closest point in root to Q
44          d ←dist(n, Q)
45          S ← PointsInRange(other, Q)
46          Update n to be itself or closest point in S
47      Return n
```

Started on	Saturday, 12 February 2022, 10:02:48 AM
State	Finished
Completed on	Saturday, 12 February 2022, 11:59:11 AM
Time taken	1 hour 56 mins
Grade	**5.85** out of 10.00 (**59**%)

Question **1**

Complete

Mark 0.50 out of 0.50

Consider the following programs

exec1.c

```
#include <unistd.h>
#include <stdio.h>
int main() {
   execl("./exec2", "./exec2", NULL);
}
```

exec2.c

```
#include <unistd.h>
#include <stdio.h>
int main() {
   execl("/bin/ls", "/bin/ls", NULL);
  printf("hello\n");
}
```

Compiled as

```
cc    exec1.c  -o exec1
cc    exec2.c  -o exec2
```

And run as

```
$ ./exec1
```

Explain the output of the above command (./exec1)

Assume that /bin/ls , i.e. the 'ls' program exists.

Select one:

- a. Execution fails as the call to execl() in exec1 fails
- b. "ls" runs on current directory
- c. Program prints hello
- d. Execution fails as the call to execl() in exec2 fails
- e. Execution fails as one exec can't invoke another exec

The correct answer is: "ls" runs on current directory

Question **2**

Complete

Mark 0.14 out of 0.50

Order the events that occur on a timer interrupt:

Select another process for execution	4
Jump to a code pointed by IDT	5
Set the context of the new process	6
Change to kernel stack of currently running process	2
Save the context of the currently running process	1
Execute the code of the new process	7
Jump to scheduler code	3

The correct answer is: Select another process for execution → 5, Jump to a code pointed by IDT → 2, Set the context of the new process → 6, Change to kernel stack of currently running process → 1, Save the context of the currently running process → 3, Execute the code of the new process → 7, Jump to scheduler code → 4

Question **3**

Not answered

Marked out of 1.00

Which parts of the xv6 code in bootasm.S bootmain.c , entry.S and in the codepath related to scheduler() and trap handling() can also be written in some other way, and still ensure that xv6 works properly?

Writing code is not necessary. You only need to comment on which part of the code could be changed to something else or written in another fashion.

Maximum two points to be written.

Question **4**

Complete

Mark 0.00 out of 1.00

Select the sequence of events that are NOT possible, assuming a non-interruptible kernel code

(Note: non-interruptible kernel code means, if the kernel code is executing, then interrupts will be disabled).

Note: A possible sequence may have some missing steps in between. An impossible sequence will will have n and n+1th steps such that n+1th step can not follow n'th step.

Select one or more:

- [ ] a. P1 running
  P1 makes system call
  system call returns
  P1 running
  timer interrupt
  Scheduler running
  P2 running

- [x] b. P1 running
  P1 makes sytem call and blocks
  Scheduler
  P2 running
  P2 makes sytem call and blocks
  Scheduler
  P3 running
  Hardware interrupt
  Interrupt unblocks P1
  Interrupt returns
  P3 running
  Timer interrupt
  Scheduler
  P1 running

- [ ] c. P1 running
  keyboard hardware interrupt
  keyboard interrupt handler running
  interrupt handler returns
  P1 running
  P1 makes sytem call
  system call returns
  P1 running
  timer interrupt
  scheduler
  P2 running

- [x] d. P1 running
  P1 makes sytem call and blocks
  Scheduler
  P2 running
  P2 makes sytem call and blocks
  Scheduler
  P1 running again

- [ ] e.
  P1 running
  P1 makes sytem call
  Scheduler
  P2 running
  P2 makes sytem call and blocks
  Scheduler
  P1 running again

- [ ] f. P1 running
  P1 makes system call
  timer interrupt
  Scheduler
  P2 running
  timer interrupt
  Scheuler
  P1 running
  P1's system call return

The correct answers are: P1 running
P1 makes sytem call and blocks
Scheduler
P2 running
P2 makes sytem call and blocks
Scheduler
P1 running again, P1 running
P1 makes system call
timer interrupt
Scheduler
P2 running
timer interrupt
Scheuler
P1 running
P1's system call return,
P1 running
P1 makes sytem call
Scheduler
P2 running
P2 makes sytem call and blocks
Scheduler
P1 running again

Question **5**

Complete

Mark 0.50 out of 0.50

Some part of the bootloader of xv6 is written in assembly while some part is written in C. Why is that so?
Select all the appropriate choices

- [x] a. The setting up of the most essential memory management infrastructure needs assembly code
- [x] b. The code in assembly is required for transition to protected mode, from real mode; after that calling convention applies, hence code can be written in C
- [ ] c. The code for reading ELF file can not be written in assembly
- [ ] d. The code in assembly is required for transition to protected mode, from real mode; but calling convention was applicable all the time

The correct answers are: The code in assembly is required for transition to protected mode, from real mode; after that calling convention applies, hence code can be written in C, The setting up of the most essential memory management infrastructure needs assembly code

Question **6**

Complete

Mark 0.50 out of 0.50

Suppose a program does a scanf() call.

Essentially the scanf does a read() system call.

This call will obviously "block" waiting for the user input.

In terms of OS data structures and execution of code, what does it mean?

Select one:

- (•) a. OS code for read() will move PCB of current process to a wait queue and call scheduler
- ( ) b. read() returns and process calls scheduler()
- ( ) c. read() will return and process will be taken to a wait queue
- ( ) d. OS code for read() will call scheduler
- ( ) e. OS code for read() will move the PCB of this process to a wait queue and return from the system call

The correct answer is: OS code for read() will move PCB of current process to a wait queue and call scheduler

Question **7**

Complete

Mark 0.40 out of 0.50

Select all the correct statements about zombie processes

Select one or more:

- [x] a. init() typically keeps calling wait() for zombie processes to get cleaned up
- [ ] b. A zombie process remains zombie forever, as there is no way to clean it up
- [ ] c. Zombie processes are harmless even if OS is up for long time
- [x] d. A process becomes zombie when it finishes, and remains zombie until parent calls wait() on it
- [ ] e. A process becomes zombie when it's parent finishes
- [ ] f. A process can become zombie if it finishes, but the parent has finished before it
- [x] g. A zombie process occupies space in OS data structures
- [x] h. If the parent of a process finishes, before the process itself, then after finishing the process is typically attached to 'init' as parent

The correct answers are: A process becomes zombie when it finishes, and remains zombie until parent calls wait() on it, A process can become zombie if it finishes, but the parent has finished before it, A zombie process occupies space in OS data structures, If the parent of a process finishes, before the process itself, then after finishing the process is typically attached to 'init' as parent, init() typically keeps calling wait() for zombie processes to get cleaned up

Question **8**

Complete

Mark 0.80 out of 1.00

Mark the statements, w.r.t. the scheduler of xv6 as True or False

True	False	
○	◉	`swtch` is a function that does not return to the caller
◉	○	`swtch` is a function that saves old context, loads new context, and returns to last EIP in the new context
◉	○	The work of selecting and scheduling a process is done only in `scheduler()` and not in `sched()`
◉	○	the control returns to `switchkvm();` after `swtch(&(c->scheduler), p->context);` in `scheduler()`
◉	○	`sched()` and `scheduler()` are co-routines
◉	○	The variable c->scheduler on first processor uses the stack allocated entry.S
○	◉	`sched()` *calls* `scheduler()` and `scheduler()` *calls* `sched()`
◉	○	The function `scheduler()` executes using the kernel-only stack
◉	○	When a process is scheduled for execution, it resumes execution in `sched()` after the call to `swtch()`
○	◉	the control returns to `mycpu()->intena = intena; ();` after `swtch(&p->context, mycpu()->scheduler);` in `sched()`

`swtch` is a function that does not return to the caller: True
`swtch` is a function that saves old context, loads new context, and returns to last EIP in the new context: True
The work of selecting and scheduling a process is done only in `scheduler()` and not in `sched()`: True
the control returns to `switchkvm();` after `swtch(&(c->scheduler), p->context);` in `scheduler()`: False
`sched()` and `scheduler()` are co-routines: True
The variable c->scheduler on first processor uses the stack allocated entry.S: True
`sched()` *calls* `scheduler()` and `scheduler()` *calls* `sched()`: False
The function `scheduler()` executes using the kernel-only stack: True
When a process is scheduled for execution, it resumes execution in `sched()` after the call to `swtch()`
: True
the control returns to `mycpu()->intena = intena; ();` after `swtch(&p->context, mycpu()->scheduler);` in `sched()`: False

Question **9**

Complete

Mark 0.29 out of 0.50

Select all the correct statements about code of bootmain() in xv6

```
void
bootmain(void)
{
  struct elfhdr *elf;
  struct proghdr *ph, *eph;
  void (*entry)(void);
  uchar* pa;

  elf = (struct elfhdr*)0x10000;  // scratch space

  // Read 1st page off disk
  readseg((uchar*)elf, 4096, 0);

  // Is this an ELF executable?
  if(elf->magic != ELF_MAGIC)
    return;  // let bootasm.S handle error

  // Load each program segment (ignores ph flags).
  ph = (struct proghdr*)((uchar*)elf + elf->phoff);
  eph = ph + elf->phnum;
  for(; ph < eph; ph++){
    pa = (uchar*)ph->paddr;
    readseg(pa, ph->filesz, ph->off);
    if(ph->memsz > ph->filesz)
      stosb(pa + ph->filesz, 0, ph->memsz - ph->filesz);
  }

  // Call the entry point from the ELF header.
  // Does not return!
  entry = (void(*)(void))(elf->entry);
  entry();
}
```

Also, inspect the relevant parts of the xv6 code. binary files, etc and run commands as you deem fit to answer this question.

- [x] a. The stosb() is used here, to fill in some space in memory with zeroes
- [ ] b. The condition if(ph->memsz > ph->filesz) is never true.
- [ ] c. The kernel file in memory is not necessarily a continuously filled in chunk, it may have holes in it.
- [x] d. The readseg finally invokes the disk I/O code using assembly instructions
- [x] e. The elf->entry is set by the linker in the kernel file and it's 8010000c
- [ ] f. The kernel ELF file contains actual physical address where particular sections of 'kernel' file should be loaded
- [ ] g. The elf->entry is set by the linker in the kernel file and it's 0x80000000
- [x] h. The kernel file gets loaded at the Physical address 0x10000 in memory.
- [ ] i. The kernel file gets loaded at the Physical address 0x10000 +0x80000000 in memory.
- [ ] j. The elf->entry is set by the linker in the kernel file and it's 0x80000000
- [ ] k. The kernel file has only two program headers

The correct answers are: The kernel file gets loaded at the Physical address 0x10000 in memory., The kernel file in memory is not necessarily a continuously filled in chunk, it may have holes in it., The elf->entry is set by the linker in the kernel file and it's 8010000c, The readseg finally invokes the disk I/O code using assembly instructions, The stosb() is used here, to fill in some space in memory with zeroes, The kernel ELF file contains actual physical address where particular sections of 'kernel' file should be loaded, The kernel file has only two program headers

Question **10**

Complete

Mark 0.50 out of 0.50

What's the trapframe in xv6?

- a. A frame of memory that contains all the trap handler code
- b. The sequence of values, including saved registers, constructed on the stack when an interrupt occurs, built by code in trapasm.S only
- c. The IDT table
- d. The sequence of values, including saved registers, constructed on the stack when an interrupt occurs, built by hardware + code in trapasm.S
- e. A frame of memory that contains all the trap handler code's function pointers
- f. The sequence of values, including saved registers, constructed on the stack when an interrupt occurs, built by hardware only
- g. A frame of memory that contains all the trap handler's addresses

The correct answer is: The sequence of values, including saved registers, constructed on the stack when an interrupt occurs, built by hardware + code in trapasm.S

Question **11**

Complete

Mark 0.50 out of 0.50

Order the sequence of events, in scheduling process P1 after process P0

Process P1 is running	6
Process P0 is running	1
context of P1 is loaded from P1's PCB	4
Control is passed to P1	5
context of P0 is saved in P0's PCB	3
timer interrupt occurs	2

The correct answer is: Process P1 is running → 6, Process P0 is running → 1, context of P1 is loaded from P1's PCB → 4, Control is passed to P1 → 5, context of P0 is saved in P0's PCB → 3, timer interrupt occurs → 2

Question **12**

Complete

Mark 0.43 out of 1.00

Select the correct statements about interrupt handling in xv6 code

- [x] a. The trapframe pointer in struct proc, points to a location on kernel stack
- [x] b. The function trap() is the called irrespective of hardware interrupt/system-call/exception
- [x] c. xv6 uses the 0x64th entry in IDT for system calls
- [x] d. All the 256 entries in the IDT are filled
- [x] e. The CS and EIP are changed only after pushing user code's SS,ESP on stack
- [ ] f. Each entry in IDT essentially gives the values of CS and EIP to be used in handling that interrupt
- [ ] g. The trapframe pointer in struct proc, points to a location on user stack
- [x] h. On any interrupt/syscall/exception the control first jumps in vectors.S
- [ ] i. The function trap() is the called only in case of hardware interrupt
- [ ] j. Before going to alltraps, the kernel stack contains upto 5 entries.
- [ ] k. The CS and EIP are changed only immediately on a hardware interrupt
- [ ] l. xv6 uses the 64th entry in IDT for system calls
- [ ] m. On any interrupt/syscall/exception the control first jumps in trapasm.S

The correct answers are: All the 256 entries in the IDT are filled, Each entry in IDT essentially gives the values of CS and EIP to be used in handling that interrupt, xv6 uses the 64th entry in IDT for system calls, On any interrupt/syscall/exception the control first jumps in vectors.S, Before going to alltraps, the kernel stack contains upto 5 entries., The trapframe pointer in struct proc, points to a location on kernel stack, The function trap() is the called irrespective of hardware interrupt/system-call/exception, The CS and EIP are changed only after pushing user code's SS,ESP on stack

Question **13**

Complete

Mark 0.45 out of 0.50

Select Yes if the mentioned element should be a part of PCB

Select No otherwise.

Yes	No	
○	◉	PID of Init
○	◉	Function pointers to all system calls
◉	○	Memory management information about that process
◉	○	PID
○	◉	Pointer to IDT
◉	○	Process context
○	◉	EIP at the time of context switch
◉	○	Process state
◉	○	Pointer to the parent process
◉	○	List of opened files

PID of Init: No
Function pointers to all system calls: No
Memory management information about that process: Yes
PID: Yes
Pointer to IDT: No
Process context: Yes
EIP at the time of context switch: Yes
Process state: Yes
Pointer to the parent process: Yes
List of opened files: Yes

Question **14**

Complete

Mark 0.10 out of 0.50

For each line of code mentioned on the left side, select the location of sp/esp that is in use

`call bootmain` in bootasm.S	The 4KB area in kernel image, loaded in memory, named as 'stack'
`jmp *%eax` in entry.S	The 4KB area in kernel image, loaded in memory, named as 'stack'
`ljmp $(SEG_KCODE<<3), $start32` in bootasm.S	0x7c00 to 0
`readseg((uchar*)elf, 4096, 0);` in bootmain.c	Immaterial as the stack is not used here
`cli` in bootasm.S	0x7c00 to 0x10000

The correct answer is: `call bootmain` in bootasm.S → 0x7c00 to 0, `jmp *%eax` in entry.S → The 4KB area in kernel image, loaded in memory, named as 'stack', `ljmp $(SEG_KCODE<<3), $start32` in bootasm.S → Immaterial as the stack is not used here, `readseg((uchar*)elf, 4096, 0);` in bootmain.c → 0x7c00 to 0, `cli` in bootasm.S → Immaterial as the stack is not used here

Question **15**

Complete

Mark 0.25 out of 0.50

The bootmain() function has this code

```
elf = (struct elfhdr*)0x10000;  // scratch space
readseg((uchar*)elf, 4096, 0);
```

Mark the statements as True or False with respect to this code.

In these statements 0x1000 is referred to as ADDRESS

True	False	
○	◉	This line loads the kernel code at ADDRESS
◉	○	This line effectively loads the ELF header and the program headers at ADDRESS
○	◉	It the value of ADDRESS is changed to a higher number (upto a limit), the program could still work
◉	○	The value ADDRESS is changed to a 0 the program could still work
◉	○	It the value of ADDRESS is changed to a lower number (upto a limit), the program could still work
○	◉	If the value of ADDRESS is changed, then the program will not work

This line loads the kernel code at ADDRESS: False
This line effectively loads the ELF header and the program headers at ADDRESS: False
It the value of ADDRESS is changed to a higher number (upto a limit), the program could still work: True
The value ADDRESS is changed to a 0 the program could still work: False
It the value of ADDRESS is changed to a lower number (upto a limit), the program could still work: True
If the value of ADDRESS is changed, then the program will not work: False

Question **16**

Complete

Mark 0.50 out of 0.50

In bootasm.S, on the line

```
ljmp    $(SEG_KCODE<<3), $start32
```

The SEG_KCODE << 3, that is shifting of 1 by 3 bits is done because

- a. While indexing the GDT using CS, the value in CS is always divided by 8
- b. The code segment is 16 bit and only upper 13 bits are used for segment number
- c. The value 8 is stored in code segment
- d. The code segment is 16 bit and only lower 13 bits are used for segment number
- e. The ljmp instruction does a divide by 8 on the first argument

The correct answer is: The code segment is 16 bit and only upper 13 bits are used for segment number

◄ Extra Reading on Linkers: A writeup by Ian Taylor (keep changing url string from 38 to 39, and so on)

Jump to...

(Code) IPC - Shm, Messages ►

Started on	Tuesday, 22 March 2022, 1:59:31 PM
State	Finished
Completed on	Tuesday, 22 March 2022, 5:23:44 PM
Time taken	3 hours 24 mins
Grade	**25.17** out of 40.00 (**63**%)

Question **1**

Partially correct

Mark 0.50 out of 1.00

Select all the correct statements about MMU and it's functionality (on a non-demand paged system)

Select one or more:

- [ ] a. Logical to physical address translations in MMU are done with specific machine instructions
- [ ] b. Illegal memory access is detected by operating system
- [x] c. The operating system interacts with MMU for every single address translation ✗
- [x] d. MMU is inside the processor ✓
- [x] e. Illegal memory access is detected in hardware by MMU and a trap is raised ✓
- [ ] f. MMU is a separate chip outside the processor
- [x] g. Logical to physical address translations in MMU are done in hardware, automatically ✓
- [ ] h. The Operating system sets up relevant CPU registers to enable proper MMU translations

Your answer is partially correct.

You have correctly selected 3.
The correct answers are: MMU is inside the processor, Logical to physical address translations in MMU are done in hardware, automatically, The Operating system sets up relevant CPU registers to enable proper MMU translations, Illegal memory access is detected in hardware by MMU and a trap is raised

Question **2**

Partially correct

Mark 0.33 out of 1.00

Select all correct statements w.r.t. Major and Minor page faults on Linux

- [x] a. Thrashing is possible only due to major page faults ✓
- [x] b. Minor page fault may occur because the page was a shared memory page ✓
- [ ] c. Minor page fault may occur because of a page fault during fork(), on code of an already running process
- [ ] d. Major page faults are likely to occur in more numbers at the beginning of the process
- [ ] e. Minor page faults are an improvement of the page buffering techniques
- [ ] f. Minor page fault may occur because the page was freed, but still tagged and available in the free page list

The correct answers are: Minor page fault may occur because the page was a shared memory page, Minor page fault may occur because of a page fault during fork(), on code of an already running process, Minor page fault may occur because the page was freed, but still tagged and available in the free page list, Major page faults are likely to occur in more numbers at the beginning of the process, Thrashing is possible only due to major page faults, Minor page faults are an improvement of the page buffering techniques

Question **3**

Partially correct

Mark 0.38 out of 1.00

Consider a demand-paging system with the following time-measured utilizations:
CPU utilization : 20%
Paging disk: 97.7%
Other I/O devices: 5%

For each of the following, indicate whether it will (or is likely to) improve CPU utilization (even if by a small amount). Explain your answers.

a. Install a faster CPU : Yes ✖

b. Install a bigger paging disk. : Yes ✖

c. Increase the degree of multiprogramming. : No ✔

d. Decrease the degree of multiprogramming. : Yes ✔

e. Install more main memory.: Yes ✔

f. Install a faster hard disk or multiple controllers with multiple hard disks. : No ✖

g. Add prepaging to the page-fetch algorithms. :
May be ✖

h. Increase the page size. : May be ✖

Question **4**

Partially correct

Mark 0.67 out of 1.00

Mark the statements as True or False, w.r.t. mmap()

True	False		
☐	☑	mmap() results in changes to page table of a process.	✖
☐	☑	MAP_SHARED leads to a mapping that is copy-on-write	✔
☐	☑	on failure mmap() returns NULL	✔
☑	☐	MAP_PRIVATE leads to a mapping that is copy-on-write	✔
☐	☑	mmap() results in changes to buffer-cache of the kernel.	✖
☑	☐	mmap() is a system call	✔
☑	☐	on failure mmap() returns (void *)-1	✔
☑	☐	mmap() can be implemented on both demand paged and non-demand paged systems.	✔
☐	☑	MAP_FIXED guarantees that the mapping is always done at the specified address	✖

mmap() results in changes to page table of a process.: True
MAP_SHARED leads to a mapping that is copy-on-write: False
on failure mmap() returns NULL: False
MAP_PRIVATE leads to a mapping that is copy-on-write: True
mmap() results in changes to buffer-cache of the kernel.: False
mmap() is a system call: True
on failure mmap() returns (void *)-1: True
mmap() can be implemented on both demand paged and non-demand paged systems.: True
MAP_FIXED guarantees that the mapping is always done at the specified address: False

Question **5**

Correct

Mark 1.00 out of 1.00

Choice of the global or local replacement strategy is a subjective choice for kernel programmers. There are advantages and disadvantages on either side. Out of the following statements, that advocate either global or local replacement strategy, select those statements that have a logically CONSISTENT argument. (That is any statement that is logically correct about either global or local replacement)

Consistent	Inconsistent		
◉☑	○✖	Local replacement can lead to under-utilisation of memory, because a process may not use all the pages allocated to it all the time.	✔
◉☑	○✖	Local replacement can be preferred when avoiding thrashing is a major concern because with local replacement and minimum number of frames allocated, a process is always able to progress and cascading inter-process page faults are avoided.	✔
◉☑	○✖	Local replacement results in more predictable per-process completion time because number of page faults can be better predicted.	✔
◉☑	○✖	Global replacement may give highly variable per process completion time because number of page faults become un-predictable.	✔
◉☑	○✖	Global replacement can be preferred when greater throughput (number of processes completing per unit time) is a concern, because each process tries to complete at the expense of others, thus leading to overall more processes completing (unless thrashing occurs).	✔

Local replacement can lead to under-utilisation of memory, because a process may not use all the pages allocated to it all the time.: Consistent
Local replacement can be preferred when avoiding thrashing is a major concern because with local replacement and minimum number of frames allocated, a process is always able to progress and cascading inter-process page faults are avoided.: Consistent
Local replacement results in more predictable per-process completion time because number of page faults can be better predicted.: Consistent
Global replacement may give highly variable per process completion time because number of page faults become un-predictable.: Consistent
Global replacement can be preferred when greater throughput (number of processes completing per unit time) is a concern, because each process tries to complete at the expense of others, thus leading to overall more processes completing (unless thrashing occurs).: Consistent

Question **6**

Partially correct

Mark 1.45 out of 2.00

W.r.t. Memory management in xv6,

xv6 uses physical memory upto 224 MB onlyMark statements True or False

True	False		
◉ ☑	○ ✖	The free page-frame are created out of nearly 222 MB	✔
◉ ✖	○ ☑	The kernel's page table given by kpgdir variable is used as stack for scheduler's context	✖
◉ ☑	○ ✖	The stack allocated in entry.S is used as stack for scheduler's context for first processor	✔
◉ ☑	○ ✖	PHYSTOP can be increased to some extent, simply by editing memlayout.h	✔
◉ ☑	○ ✖	The process's address space gets mapped on frames, obtained from ~2MB:224MB range	✔
◉ ✖	○ ☑	The switchkvm() call in scheduler() changes CR3 to use page directory of new process	✖
◉ ☑	○ ✖	xv6 uses physical memory upto 224 MB only	✔
◉ ☑	○ ✖	The switchkvm() call in scheduler() is invoked after control comes to it from sched(), thus demanding execution in kernel's context	✔
◉ ☑	○ ✖	The switchkvm() call in scheduler() changes CR3 to use page directory kpgdir	✔
◉ ☑	○ ✖	The kernel code and data take up less than 2 MB space	✔
◉ ✖	○ ☑	The switchkvm() call in scheduler() is invoked after control comes to it from swtch() scheduler(), thus demanding execution in new process's context	✖

The free page-frame are created out of nearly 222 MB: True
The kernel's page table given by kpgdir variable is used as stack for scheduler's context: False
The stack allocated in entry.S is used as stack for scheduler's context for first processor: True
PHYSTOP can be increased to some extent, simply by editing memlayout.h: True
The process's address space gets mapped on frames, obtained from ~2MB:224MB range: True
The switchkvm() call in scheduler() changes CR3 to use page directory of new process: False
xv6 uses physical memory upto 224 MB only: True
The switchkvm() call in scheduler() is invoked after control comes to it from sched(), thus demanding execution in kernel's context: True
The switchkvm() call in scheduler() changes CR3 to use page directory kpgdir: True
The kernel code and data take up less than 2 MB space: True
The switchkvm() call in scheduler() is invoked after control comes to it from swtch() scheduler(), thus demanding execution in new process's context: False

Question **7**

Partially correct

Mark 0.35 out of 1.00

Select all the correct statements about linking and loading.

Select one or more:

- [ ] a. Continuous memory management schemes can support dynamic linking and dynamic loading.
- [ ] b. Continuous memory management schemes can support static linking and dynamic loading. (may be inefficiently)
- [x] c. Continuous memory management schemes can support static linking and static loading. (may be inefficiently) ✔
- [ ] d. Loader is last stage of the linker program
- [ ] e. Dynamic linking and loading is not possible without demand paging or demand segmentation.
- [x] f. Dynamic linking essentially results in relocatable code. ✔
- [x] g. Static linking leads to non-relocatable code ✖
- [x] h. Loader is part of the operating system ✔
- [ ] i. Dynamic linking is possible with continous memory management, but variable sized partitions only.

Your answer is partially correct.

You have correctly selected 3.
The correct answers are: Continuous memory management schemes can support static linking and static loading. (may be inefficiently), Continuous memory management schemes can support static linking and dynamic loading. (may be inefficiently), Dynamic linking essentially results in relocatable code., Loader is part of the operating system, Dynamic linking and loading is not possible without demand paging or demand segmentation.

Question **8**

Partially correct

Mark 0.83 out of 1.00

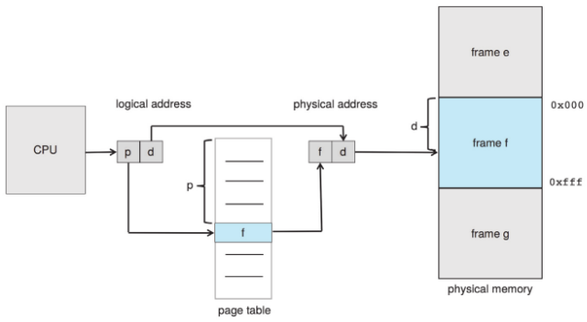


Figure 9.8 Paging hardware.

Mark the statements as True or False, w.r.t. the above diagram (note that the diagram does not cover all details of what actually happens!)

True	False		
◉ ☑	○ ✖	The combining of f and d is done by MMU	✔
◉ ☑	○ ✖	The split of logical address into p and d is done by MMU	✔
◉ ☑	○ ✖	The page table is in physical memory and must be continuous	✔
◉ ☑	○ ✖	The logical address issued by CPU is the same one generated by compiler	✔
◉ ✖	○ ☑	Using the offset d in the physical page-frame is done by MMU	✖
○ ✖	◉ ☑	There are total 3 memory references in this diagram	✔

The combining of f and d is done by MMU: True
The split of logical address into p and d is done by MMU: True
The page table is in physical memory and must be continuous: True
The logical address issued by CPU is the same one generated by compiler: True
Using the offset d in the physical page-frame is done by MMU: False
There are total 3 memory references in this diagram: False

Question **9**

Correct

Mark 1.00 out of 1.00

If one thread opens a file with read privileges then

Select one:

- a. any other thread cannot read from that file
- b. none of these
- c. other threads in the another process can also read from that file
- d. other threads in the same process can also read from that file ✔

Your answer is correct.

The correct answer is: other threads in the same process can also read from that file

Question **10**

Correct

Mark 2.00 out of 2.00

Consider the reference string
6 4 2 0 1 2 6 9 2 0 5

If the number of page frames is 3, then total number of page faults (including initial), using FIFO replacement is:

Answer: 10 ✔

#6# 6,4# 6,4,2 #0,4,2# 0,1,2 #0,1,6 #9,1,6# 9,2,6# 9,2,0 #5,2,0

The correct answer is: 10

Question **11**

Correct

Mark 1.00 out of 1.00

The data structure used in kalloc() and kfree() in xv6 is

- a. Singly linked NULL terminated list ✔
- b. Double linked NULL terminated list
- c. Doubly linked circular list
- d. Singly linked circular list

Your answer is correct.

The correct answer is: Singly linked NULL terminated list

Question **12**

Not answered

Marked out of 1.00

Suppose a kernel uses a buddy allocator. The smallest chunk that can be allocated is of size 32 bytes. One bit is used to track each such chunk, where 1 means allocated and 0 means free. The chunk looks like this as of now:

10011010

Now, there is a request for a chunk of 50 bytes.

After this allocation, the bitmap, indicating the status of the buddy allocator will be

Answer: ✖

The correct answer is: 11111010

Question **13**

Correct

Mark 1.00 out of 1.00

Given that a kernel has 1000 KB of total memory, and holes of sizes (in that order) 300 KB, 200 KB, 100 KB, 250 KB. For each of the requests on the left side, match it with the chunk chosen using the specified algorithm.

Consider each request as first request.

200 KB, first fit	300 KB	✔
150 KB, first fit	300 KB	✔
150 KB, best fit	200 KB	✔
220 KB, best fit	250 KB	✔
50 KB, worst fit	300 KB	✔
100 KB, worst fit	300 KB	✔

The correct answer is: 200 KB, first fit → 300 KB, 150 KB, first fit → 300 KB, 150 KB, best fit → 200 KB, 220 KB, best fit → 250 KB, 50 KB, worst fit → 300 KB, 100 KB, worst fit → 300 KB

Question **14**

Incorrect

Mark 0.00 out of 1.00

Given below is a sequence of reference bits on pages before the second chance algorithm runs. Before the algorithm runs, the counter is at the page marked (x). Write the sequence of reference bits after the second chance algorithm has executed once. In the answer write PRECISELY one space BETWEEN each number and do not mention (x).

0 0 1(x) 1 0 1 1

Answer: 0 0 1 1 0 0 0 ✖

The correct answer is: 0 0 0 0 0 1 1

Question **15**

Partially correct

Mark 0.10 out of 2.00

Order the following events, in the creation of init() process in xv6:

1. ✖ name of process "/init" is copied in struct proc
2. ✖ kernel memory mappings are created for initcode
3. ✖ page table mappings of 'initcode' are replaced by makpings of 'init'
4. ✖ Stack is allocated for "/init" process
5. ✖ function pointer from syscalls[] array is invoked
6. ✔ empty struct proc is obtained for initcode
7. ✖ kernel stack is allocated for initcode process
8. ✖ code is set to start in forkret() when process gets scheduled
9. ✖ initcode process runs
10. ✖ memory mappings are created for "/init" process
11. ✖ trap() runs
12. ✖ trapframe and context pointers are set to proper location
13. ✖ initcode is selected by scheduler for execution
14. ✖ userinit() is called
15. ✖ Arguments on setup on process stack for /init
16. ✖ initcode process is set to be runnable
17. ✖ sys_exec runs
18. ✖ values are set in the trapframe of initcode
19. ✖ the header of "/init" ELF file is ready by kernel
20. ✖ initcode calls exec system call

Your answer is partially correct.

Grading type: Relative to the next item (including last)

Grade details: 1 / 20 = 5%

Here are the scores for each item in this response:

1. 0 / 1 = 0%
2. 0 / 1 = 0%
3. 0 / 1 = 0%
4. 0 / 1 = 0%
5. 0 / 1 = 0%
6. 1 / 1 = 100%
7. 0 / 1 = 0%
8. 0 / 1 = 0%
9. 0 / 1 = 0%
10. 0 / 1 = 0%
11. 0 / 1 = 0%
12. 0 / 1 = 0%
13. 0 / 1 = 0%
14. 0 / 1 = 0%
15. 0 / 1 = 0%
16. 0 / 1 = 0%
17. 0 / 1 = 0%
18. 0 / 1 = 0%
19. 0 / 1 = 0%
20. 0 / 1 = 0%

The correct order for these items is as follows:

1. userinit() is called
2. empty struct proc is obtained for initcode

3. kernel stack is allocated for initcode process
4. trapframe and context pointers are set to proper location
5. code is set to start in forkret() when process gets scheduled
6. kernel memory mappings are created for initcode
7. values are set in the trapframe of initcode
8. initcode process is set to be runnable
9. initcode is selected by scheduler for execution
10. initcode process runs
11. initcode calls exec system call
12. trap() runs
13. function pointer from syscalls[] array is invoked
14. sys_exec runs
15. the header of "/init" ELF file is ready by kernel
16. memory mappings are created for "/init" process
17. Stack is allocated for "/init" process
18. Arguments on setup on process stack for /init
19. name of process "/init" is copied in struct proc
20. page table mappings of 'initcode' are replaced by makpings of 'init'

Question **16**

Correct

Mark 1.00 out of 1.00

Select all the correct statements about process states.

Note that in this question you lose marks for every incorrect choice that you make, proportional to actual number of incorrect choices.

- [x] a. Process state can be implemented as just a number ✔
- [ ] b. A process becomes ZOMBIE when another process bites into it's memory
- [ ] c. Process state is stored in the processor
- [ ] d. The scheduler can change state of a process from RUNNALBE to RUNNING and vice-versa
- [ ] e. Process state is changed only by interrupt handlers
- [ ] f. Process state is implemented as a string
- [x] g. The scheduler can change state of a process from RUNNALBE to RUNNING ✔
- [x] h. Process state is stored in the PCB ✔
- [x] i. A process becomes ZOMBIE when it calls exit() ✔

Your answer is correct.

The correct answers are: Process state is stored in the PCB, Process state can be implemented as just a number, The scheduler can change state of a process from RUNNALBE to RUNNING, A process becomes ZOMBIE when it calls exit()

Question **17**

Partially correct

Mark 0.50 out of 1.00

For the reference string
3 4 3 5 2

using LRU replacement policy for pages,

consider the number of page faults for 2, 3 and 4 page frames.
Select the most correct statement.

Select one:

- a. LRU will never exhibit Balady's anomaly
- b. Exhibit Balady's anomaly between 3 and 4 frames
- c. This example does not exhibit Balady's anomaly ☑
- d. Exhibit Balady's anomaly between 2 and 3 frames

Your answer is partially correct.

The correct answer is: LRU will never exhibit Balady's anomaly

Question **18**

Partially correct

Mark 0.67 out of 1.00

Select the most common causes of use of IPC by processes

- a. Breaking up a large task into small tasks and speeding up computation, on multiple core machines ✔
- b. Sharing of information of common interest
- c. More security checks
- d. Get the kernel performance statistics
- e. More modular code ✔

The correct answers are: Sharing of information of common interest, Breaking up a large task into small tasks and speeding up computation, on multiple core machines, More modular code

Question **19**

Partially correct

Mark 0.50 out of 1.00

Mark whether the given sequence of events is possible or not-possible. Also, select the reason for your answer.

For each sequence it's a not-possible sequence if some important event is not mentioned in the sequence.

Assume that the kernel code is non-interruptible and uniprocessor system.

```
Process P1, user code executing
Timer interrupt
Context changes to kernel context
Generic interrupt handler runs
Generic interrupt handler calls Scheduler
Scheduler selects P2 for execution
After scheduler, Process P2 user code executing
```

This sequence of events is: not-possible ✔

Because

Timer interrupt is not possible as kernel is non-interruptible ✖

Question **20**

Partially correct

Mark 0.25 out of 1.00

Select all the correct statements about signals

Select one or more:

- [ ] a. SIGKILL definitely kills a process because it's code runs in kernel mode of CPU
- [x] b. The signal handler code runs in kernel mode of CPU ✗
- [x] c. Signals are delivered to a process by kernel ✓
- [ ] d. A signal handler can be invoked asynchronously or synchronously depending on signal type
- [ ] e. Signal handlers once replaced can't be restored
- [x] f. SIGKILL definitely kills a process because it can't be caught or ignored, and it's default action terminates the process ✓
- [ ] g. The signal handler code runs in user mode of CPU
- [ ] h. Signals are delivered to a process by another process

Your answer is partially correct.

You have correctly selected 2.
The correct answers are: Signals are delivered to a process by kernel, A signal handler can be invoked asynchronously or synchronously depending on signal type, The signal handler code runs in user mode of CPU, SIGKILL definitely kills a process because it can't be caught or ignored, and it's default action terminates the process

Question **21**

Partially correct

Mark 1.56 out of 2.00

Match the description of a memory management function with the name of the function that provides it, in xv6

Setup and load the user page table for initcode process	inituvm() ✓
Switch to user page table	switchuvm() ✓
Switch to kernel page table	switchkvm() ✓
Create a copy of the page table of a process	copyuvm() ✓
Load contents from ELF into pages after allocating the pages first	loaduvm() ✗
Copy the code pages of a process	copyuvm() ✗
Load contents from ELF into existing pages	loaduvm() ✓
Mark the page as in-accessible	clearpteu() ✓
setup the kernel part in the page table	setupkvm() ✓

The correct answer is: Setup and load the user page table for initcode process → inituvm(), Switch to user page table → switchuvm(), Switch to kernel page table → switchkvm(), Create a copy of the page table of a process → copyuvm(), Load contents from ELF into pages after allocating the pages first → No such function, Copy the code pages of a process → No such function, Load contents from ELF into existing pages → loaduvm(), Mark the page as in-accessible → clearpteu(), setup the kernel part in the page table → setupkvm()

Question **22**

Partially correct

Mark 0.63 out of 1.00

Mark the statements as True or False, w.r.t. passing of arguments to system calls in xv6 code.

True	False		
◉ ☑	○ ✖	The arguments are accessed in the kernel code using esp on the trapframe.	✔
◉ ☑	○ ✖	Integer arguments are copied from user memory to kernel memory using argint()	✔
◉ ☑	○ ✖	String arguments are NOT copied in kernel memory, but just pointed to by a kernel memory pointer	✔
○ ✖	◉ ☑	The arguments to system call are copied to kernel stack in trapasm.S	✔
◉ ✖	○ ☑	Integer arguments are stored in eax, ebx, ecx, etc. registers	✖
◉ ✖	○ ☑	String arguments are first copied to trapframe and then from trapframe to kernel's other variables.	✖
○ ☑	◉ ✖	The arguments to system call originally reside on process stack.	✖
◉ ☑	○ ✖	The functions like argint(), argstr() make the system call arguments available in the kernel.	✔

The arguments are accessed in the kernel code using esp on the trapframe.: True
Integer arguments are copied from user memory to kernel memory using argint(): True
String arguments are NOT copied in kernel memory, but just pointed to by a kernel memory pointer: True
The arguments to system call are copied to kernel stack in trapasm.S: False
Integer arguments are stored in eax, ebx, ecx, etc. registers: False
String arguments are first copied to trapframe and then from trapframe to kernel's other variables.: False
The arguments to system call originally reside on process stack.: True
The functions like argint(), argstr() make the system call arguments available in the kernel.: True

Question **23**

Partially correct

Mark 0.50 out of 1.00

Map the functionality/use with function/variable in xv6 code.

Setup kernel part of a page table, mapping kernel code, data, read-only data, I/O space, devices	mappages() ✗
Create page table entries for a given range of virtual and physical addresses; including page directory entries if needed	walkpgdir() ✗
Array listing the kernel memory mappings, to be used by setupkvm()	kmap[] ✓
Return address of page table entry in a given page directory, for a given virtual address; creates page table if necessary	walkpgdir() ✓
Setup kernel part of a page table, and switch to that page table	setupkvm() ✗
return a free page, if available; 0, otherwise	kalloc() ✓

Your answer is partially correct.

You have correctly selected 3.
The correct answer is: Setup kernel part of a page table, mapping kernel code, data, read-only data, I/O space, devices → setupkvm(), Create page table entries for a given range of virtual and physical addresses; including page directory entries if needed → mappages(), Array listing the kernel memory mappings, to be used by setupkvm() → kmap[], Return address of page table entry in a given page directory, for a given virtual address; creates page table if necessary → walkpgdir(), Setup kernel part of a page table, and switch to that page table → kvmalloc(), return a free page, if available; 0, otherwise → kalloc()

Question **24**

Partially correct

Mark 0.60 out of 1.00

Mark statements True/False w.r.t. change of states of a process. Note that a statement is true only if the claim and argument both are true.

Reference: The process state diagram (and your understanding of how kernel code works). Note - the diagram does not show zombie state!

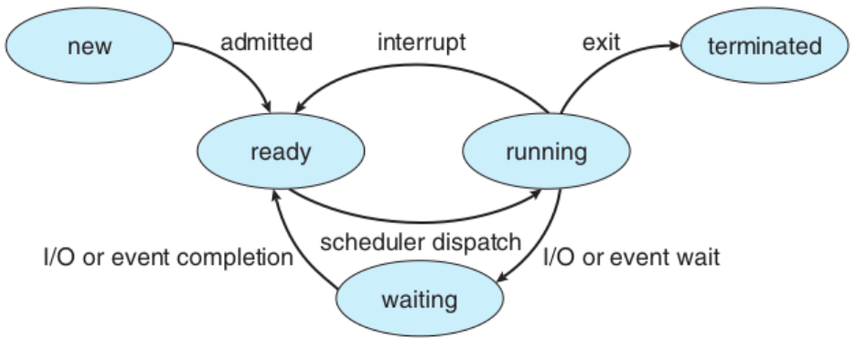


Figure 3.2 Diagram of process state.

True	False		
◉☑	○✖	A process in WAITING state can not become RUNNING because the event it's waiting for has not occurred and it has not been moved to ready queue yet	✔
◉✖	○☑	A process in READY state can not go to WAITING state because the resource on which it will WAIT will not be in use when process is in READY state.	✖
◉☑	○✖	Only a process in READY state is considered by scheduler	✔
◉✖	○☑	A process only in RUNNING state can become TERMINATED because scheduler moves it to ZOMBIE state first	✖
◉☑	○✖	Every forked process has to go through ZOMBIE state, at least for a small duration.	✔

A process in WAITING state can not become RUNNING because the event it's waiting for has not occurred and it has not been moved to ready queue yet: True
A process in READY state can not go to WAITING state because the resource on which it will WAIT will not be in use when process is in READY state.: False
Only a process in READY state is considered by scheduler: True
A process only in RUNNING state can become TERMINATED because scheduler moves it to ZOMBIE state first: False
Every forked process has to go through ZOMBIE state, at least for a small duration.: True

Question **25**

Partially correct

Mark 0.71 out of 1.00

After virtual memory is implemented

(select T/F for each of the following)One Program's size can be larger than physical memory size

True	False		
○ ✖	○ ☑	Virtual addresses become available to executing process	✖
○ ☑	○ ✖	Cumulative size of all programs can be larger than physical memory size	✔
○ ✖	○ ☑	Virtual access to memory is granted to all processes	✖
○ ☑	○ ✖	Relatively less I/O may be possible during process execution	✔
○ ☑	○ ✖	Code need not be completely in memory	✔
○ ☑	○ ✖	Logical address space could be larger than physical address space	✔
○ ☑	○ ✖	One Program's size can be larger than physical memory size	✔

Virtual addresses become available to executing process: False
Cumulative size of all programs can be larger than physical memory size: True
Virtual access to memory is granted to all processes: False
Relatively less I/O may be possible during process execution: True
Code need not be completely in memory: True
Logical address space could be larger than physical address space: True
One Program's size can be larger than physical memory size: True

Question **26**

Partially correct

Mark 0.50 out of 1.00

W.r.t. xv6 code, match the state of a process with a code that sets the state

ZOMBIE	exit(), called by an interrupt handler	✖
SLEEPING	sleep(), called by any process blocking itself	✔
RUNNABLE	exit(), called by process itself	✖
UNUSED	wait() called by the exiting process itself	✖
EMBRYO	fork()->allocproc() before setting up the UVM	✔
RUNNING	scheduler()	✔

The correct answer is: ZOMBIE → exit(), called by process itself, SLEEPING → sleep(), called by any process blocking itself, RUNNABLE → wakeup(), called by an interrupt handler, UNUSED → wait(), called by parent process, EMBRYO → fork()->allocproc() before setting up the UVM, RUNNING → scheduler()

Question **27**

Partially correct

Mark 0.50 out of 1.00

Select the correct statements about interrupt handling in xv6 code

- [ ] a. The trapframe pointer in struct proc, points to a location on kernel stack
- [ ] b. On any interrupt/syscall/exception the control first jumps in vectors.S
- [ ] c. Each entry in IDT essentially gives the values of CS and EIP to be used in handling that interrupt
- [ ] d. The function trap() is the called only in case of hardware interrupt
- [x] e. The function trap() is the called irrespective of hardware interrupt/system-call/exception ✔
- [x] f. All the 256 entries in the IDT are filled ✔
- [ ] g. The CS and EIP are changed only immediately on a hardware interrupt
- [ ] h. The trapframe pointer in struct proc, points to a location on user stack
- [ ] i. Before going to alltraps, the kernel stack contains upto 5 entries.
- [x] j. xv6 uses the 64th entry in IDT for system calls ✔
- [ ] k. xv6 uses the 0x64th entry in IDT for system calls
- [x] l. The CS and EIP are changed only after pushing user code's SS,ESP on stack ✔
- [ ] m. On any interrupt/syscall/exception the control first jumps in trapasm.S

Your answer is partially correct.

You have correctly selected 4.
The correct answers are: All the 256 entries in the IDT are filled, Each entry in IDT essentially gives the values of CS and EIP to be used in handling that interrupt, xv6 uses the 64th entry in IDT for system calls, On any interrupt/syscall/exception the control first jumps in vectors.S, Before going to alltraps, the kernel stack contains upto 5 entries., The trapframe pointer in struct proc, points to a location on kernel stack, The function trap() is the called irrespective of hardware interrupt/system-call/exception, The CS and EIP are changed only after pushing user code's SS,ESP on stack

Question **28**

Partially correct

Mark 0.25 out of 1.00

Select the correct points of comparison between POSIX and System V shared memory.

- [ ] a. POSIX shared memory is "thread safe", System V is not
- [x] b. POSIX shared memory is newer than System V shared memory ✔
- [ ] c. System V is more prevalent than POSIX even today
- [ ] d. POSIX allows giving name to shared memory, System V does not

The correct answers are: POSIX shared memory is newer than System V shared memory, POSIX shared memory is "thread safe", System V is not, POSIX allows giving name to shared memory, System V does not, System V is more prevalent than POSIX even today

Question **29**

Partially correct

Mark 0.50 out of 1.00

For each function/code-point, select the status of segmentation setup in xv6

entry.S	gdt setup with 3 entries, at start32 symbol of bootasm.S ✔
kvmalloc() in main()	gdt setup with 3 entries, at start32 symbol of bootasm.S ✔
after startothers() in main()	gdt setup with 5 entries (0 to 4) on one processor ✖
after seginit() in main()	gdt setup with 5 entries (0 to 4) on one processor ✔
bootasm.S	gdt setup with 3 entries, right from first line of code of bootloader ✖
bootmain()	gdt setup with 3 entries, right from first line of code of bootloader ✖

Your answer is partially correct.

You have correctly selected 3.
The correct answer is: entry.S → gdt setup with 3 entries, at start32 symbol of bootasm.S, kvmalloc() in main() → gdt setup with 3 entries, at start32 symbol of bootasm.S, after startothers() in main() → gdt setup with 5 entries (0 to 4) on all processors, after seginit() in main() → gdt setup with 5 entries (0 to 4) on one processor, bootasm.S → gdt setup with 3 entries, at start32 symbol of bootasm.S, bootmain() → gdt setup with 3 entries, at start32 symbol of bootasm.S

Question **30**

Correct

Mark 2.00 out of 2.00

For the reference string
3 4 3 5 2

using FIFO replacement policy for pages,

consider the number of page faults for 2, 3 and 4 page frames.
Select the correct statement.

Select one:

- a. Do not exhibit Balady's anomaly ✔
- b. Exhibit Balady's anomaly between 2 and 3 frames
- c. Exhibit Balady's anomaly between 3 and 4 frames

Your answer is correct.

The correct answer is: Do not exhibit Balady's anomaly

Question **31**

Correct

Mark 1.00 out of 1.00

Consider a computer system with a 32-bit logical address and 4- KB page size. The system supports up to 512 MB of physical memory. How many entries are there in each of the following?

Write answer as a decimal number.

A conventional, single-level page table:

1048576 ✔

An inverted page table:

131072 ✔

Question **32**

Partially correct

Mark 0.70 out of 1.00

Mark the statements about named and un-named pipes as True or False

True	False		
◉ ☑	○ ✖	Un-named pipes can be used for communication between only "related" processes, if the common ancestor created it.	✔
◉ ☑	○ ✖	Un-named pipes are inherited by a child process from parent.	✔
◉ ✖	○ ☑	Named pipes can be used for communication between only "related" processes.	✖
◉ ☑	○ ✖	Both types of pipes are an extension of the idea of "message passing".	✔
◉ ☑	○ ✖	Named pipes can exist beyond the life-time of processes using them.	✔
○ ✖	◉ ☑	The buffers for named-pipe are in process-memory while the buffers for the un-named pipe are in kernel memory.	✔
○ ✖	◉ ☑	A named pipe has a name decided by the kernel.	✔
○ ☑	◉ ✖	Both types of pipes provide FIFO communication.	✖
◉ ✖	○ ☑	The pipe() system call can be used to create either a named or un-named pipe.	✖
◉ ☑	○ ✖	Named pipe exists as a file	✔

Un-named pipes can be used for communication between only "related" processes, if the common ancestor created it.: True
Un-named pipes are inherited by a child process from parent.: True
Named pipes can be used for communication between only "related" processes.: False
Both types of pipes are an extension of the idea of "message passing".: True
Named pipes can exist beyond the life-time of processes using them.: True
The buffers for named-pipe are in process-memory while the buffers for the un-named pipe are in kernel memory.: False
A named pipe has a name decided by the kernel.: False
Both types of pipes provide FIFO communication.: True
The pipe() system call can be used to create either a named or un-named pipe.: False
Named pipe exists as a file: True

Question **33**

Partially correct

Mark 0.60 out of 1.00

Select all the correct statements w.r.t user and kernel threads

Select one or more:

- [ ] a. A process may not block in many-one model, if a thread makes a blocking system call
- [x] b. one-one model increases kernel's scheduling load ✔
- [x] c. all three models, that is many-one, one-one, many-many , require a user level thread library ✔
- [x] d. A process blocks in many-one model even if a single thread makes a blocking system call ✔
- [ ] e. one-one model can be implemented even if there are no kernel threads
- [ ] f. many-one model can be implemented even if there are no kernel threads
- [ ] g. many-one model gives no speedup on multicore processors

Your answer is partially correct.

You have correctly selected 3.

The correct answers are: many-one model can be implemented even if there are no kernel threads, all three models, that is many-one, one-one, many-many , require a user level thread library, one-one model increases kernel's scheduling load, many-one model gives no speedup on multicore processors, A process blocks in many-one model even if a single thread makes a blocking system call

Question **34**

Partially correct

Mark 0.60 out of 1.00

Mark the statements as True or False, w.r.t. thrashing

True	False		
✖	☑	Thrashing occurs because some process is doing lot of disk I/O.	✖
✖	☑	mmap() solves the problem of thrashing.	✖
☑	✖	Thrashing can be limited if local replacement is used.	✔
☑	✖	Thrashing occurs when the total size of all processe's locality exceeds total memory size.	✔
☑	✖	Thrashing is particular to demand paging systems, and does not apply to pure paging systems.	✔
☑	✖	Processes keep changing their locality of reference, and a high rate of page faults occur when they are changing the locality.	✔
☑	✖	During thrashing the CPU is under-utilised as most time is spent in I/O	✔
☑	✖	The working set model is an attempt at approximating the locality of a process.	✔
✖	☑	Processes keep changing their locality of reference, and least number of page faults occur when they are changing the locality.	✖
✖	☑	Thrashing can occur even if entire memory is not in use.	✖

Thrashing occurs because some process is doing lot of disk I/O.: False
mmap() solves the problem of thrashing.: False
Thrashing can be limited if local replacement is used.: True
Thrashing occurs when the total size of all processe's locality exceeds total memory size.: True
Thrashing is particular to demand paging systems, and does not apply to pure paging systems.: True
Processes keep changing their locality of reference, and a high rate of page faults occur when they are changing the locality.: True
During thrashing the CPU is under-utilised as most time is spent in I/O: True
The working set model is an attempt at approximating the locality of a process.: True
Processes keep changing their locality of reference, and least number of page faults occur when they are changing the locality.: False
Thrashing can occur even if entire memory is not in use.: False

Question **35**

Correct

Mark 1.00 out of 1.00

The complete range of virtual addresses (after main() in main.c is over), from which the free pages used by kalloc() and kfree() is derived,are:

- a. end, P2V(PHYSTOP) ✔
- b. end, PHYSTOP
- c. P2V(end), P2V(PHYSTOP)
- d. P2V(end), PHYSTOP
- e. end, 4MB
- f. end, P2V(4MB + PHYSTOP)
- g. end, (4MB + PHYSTOP)

Your answer is correct.

The correct answer is: end, P2V(PHYSTOP)

◄ (Optional Assignment) lseek system call in xv6

Jump to...

Feedback on Quiz-2 ►

13

Randomized Dictionary Structures

C. Pandu Rangan
Indian Institute of Technology, Madras

13.1 Introduction

In the last couple of decades, there has been a tremendous growth in using randomness as a powerful source of computation. Incorporating randomness in computation often results in a much simpler and more easily implementable algorithms. A number of problem domains, ranging from sorting to stringology, from graph theory to computational geometry, from parallel processing system to ubiquitous internet, have benefited from randomization in terms of newer and elegant algorithms. In this chapter we shall see how randomness can be used as a powerful tool for designing simple and efficient data structures. Solving a real-life problem often involves manipulating complex data objects by variety of operations. We use abstraction to arrive at a mathematical model that represents the real-life objects and convert the real-life problem into a computational problem working on the mathematical entities specified by the model. Specifically, we define *Abstract Data Type (ADT)* as a mathematical model together with a set of operations defined on the entities of the model. Thus, an algorithm for a computational problem will be expressed in terms of the steps involving the corresponding ADT operations. In order to arrive at a computer based implementation of the algorithm, we need to proceed further taking a closer look at the possibilities of implementing the ADTs. As programming languages support only a very small number of built-in types, any ADT that is not a built-in type must be represented in terms of the elements from built-in type and this is where the data structure plays a critical role. One major goal in the design of data structure is to render the operations of the ADT as efficient as possible. Traditionally, data structures were designed to minimize the worst-case costs of the ADT operations. When the worst-case efficient data structures turn out to be too complex and cumbersome to implement, we naturally explore alternative

design goals. In one of such design goals, we seek to minimize the total cost of a sequence of operations as opposed to the cost of individual operations. Such data structures are said to be designed for minimizing the amortized costs of operations. Randomization provides yet another avenue for exploration. Here, the goal will be to limit the expected costs of operations and ensure that costs do not exceed certain threshold limits with overwhelming probability.

In this chapter we discuss about the *Dictionary* ADT which deals with sets whose elements are drawn from a fixed universe U and supports operations such as *insert*, *delete* and *search*. Formally, we assume a linearly ordered universal set U and for the sake of concreteness we assume U to be the set of all integers. At any time of computation, the *Dictionary* deals only with a finite subset of U. We shall further make a simplifying assumption that we deal only with sets with distinct values. That is, we never handle a multiset in our structure, though, with minor modifications, our structures can be adjusted to handle multisets containing multiple copies of some elements. With these remarks, we are ready for the specification of the *Dictionary* ADT.

DEFINITION 13.1 [Dictionary ADT] Let U be a linearly ordered universal set and S denote a finite subset of U. The Dictionary ADT, defined on the class of finite subsets of U, supports the following operations.
Insert (x, S) : For an $x \in U, S \subset U$, generate the set $S \bigcup \{x\}$.
Delete (x, S) : For an $x \in U, S \subset U$, generate the set $S - \{x\}$.
Search (x, S) : For an $x \in U, S \subset U$, return TRUE if $x \in S$ and return FALSE if $x \notin S$.

Remark : When the universal set is evident in a context, we will not explicitly mention it in the discussions. Notice that we work with sets and not multisets. Thus, *Insert (x,S)* does not produce new set when x is in the set already. Similarly *Delete (x, S)* does not produce a new set when $x \notin S$.

Due to its fundamental importance in a host of applications ranging from compiler design to data bases, extensive studies have been done in the design of data structures for dictionaries. Refer to Chapters 3 and 10 for data structures for dictionaries designed with the worst-case costs in mind, and Chapter 12 of this handbook for a data structure designed with amortized cost in mind. In Chapter 15 of this book, you will find an account of *B-Trees* which aim to minimize the disk access. All these structures, however, are deterministic. In this sequel, we discuss two of the interesting randomized data structures for Dictionaries. Specifically

- We describe a data structure called *Skip Lists* and present a comprehensive probabilistic analysis of its performance.
- We discuss an interesting randomized variation of a search tree called *Randomized Binary Search Tree* and compare and contrast the same with other competing structures.

13.2 Preliminaries

In this section we collect some basic definitions, concepts and the results on randomized computations and probability theory. We have collected only the materials needed for the topics discussed in this chapter. For a more comprehensive treatment of randomized algorithms, refer to the book by Motwani and Raghavan [9].

13.2.1 Randomized Algorithms

Every computational step in an execution of a *deterministic algorithm* is uniquely determined by the set of all steps executed prior to this step. However, in a *randomized algorithm*, the choice of the next step may not be entirely determined by steps executed previously; the choice of next step might depend on the outcome of a random number generator. Thus, several execution sequences are possible even for the same input. Specifically, when a randomized algorithm is executed several times, even on the same input, the running time may vary from one execution to another. In fact, the running time is a random variable depending on the random choices made during the execution of the algorithm. When the running time of an algorithm is a random variable, the traditional worst case complexity measure becomes inappropriate. In fact, the quality of a randomized algorithm is judged from the statistical properties of the random variable representing the running time. Specifically, we might ask for bounds for the expected running time and bounds beyond which the running time may exceed only with negligible probability. In other words, for the randomized algorithms, there is no bad input; we may perhaps have an *unlucky* execution.
The type of randomized algorithms that we discuss in this chapter is called *Las Vegas* type algorithms. A *Las Vegas* algorithm always terminates with a correct answer although the running time may be a random variable exhibiting wide variations. There is another important class of randomized algorithms, called *Monte Carlo* algorithms, which have fixed running time but the output may be erroneous. We will not deal with *Monte Carlo* algorithms as they are not really appropriate for basic building blocks such as data structures. We shall now define the notion of efficiency and complexity measures for *Las Vegas* type randomized algorithms.

Since the running time of a *Las Vegas* randomized algorithm on any given input is a random variable, besides determining the expected running time it is desirable to show that the running time does not exceed certain threshold value with very high probability. Such threshold values are called *high probability bounds* or *high confidence bounds*. As is customary in algorithmics, we express the estimation of the expected bound or the high-probability bound as a function of the size of the input. We interpret an execution of a *Las Vegas* algorithm as a *failure* if the running time of the execution exceeds the expected running time or the high-confidence bound.

DEFINITION 13.2 [Confidence Bounds] Let α, β and c be positive constants. A randomized algorithm A requires resource bound $f(n)$ with

1. $n-$*exponential* probability or very high probability, if for any input of size n, the amount of the resource used by A is at most $\alpha f(n)$ with probability $1 - O(\beta^{-n})$, $\beta > 1$. In this case $f(n)$ is called a *very high confidence bound*.
2. n − *polynomial* probability or high probability, if for any input of size n, the

amount of the resource used by A is at most $\alpha f(n)$ with probability $1 - O(n^{-c})$. In this case $f(n)$ is called a *high confidence bound*.

3. $n - log$ probability or very good probability, if for any input of size n, the amount of the resource used by A is at most $\alpha f(n)$ with probability $1 - O((\log n)^{-c})$. In this case $f(n)$ is called a *very good confidence bound*.
4. $high - constant$ probability, if for any input of size n, the amount of the resource used by A is at most $\alpha f(n)$ with probability $1 - O(\beta^{-\alpha})$, $\beta > 1$.

The practical significance of this definition can be understood from the following discussions. For instance, let A be a *Las Vegas* type algorithm with $f(n)$ as a high confidence bound for its running time. As noted before, the actual running time $T(n)$ may vary from one execution to another but the definition above implies that, for any execution, on any input, $Pr(T(n) > f(n)) = O(n^{-c})$. Even for modest values of n and c, this bound implies an extreme rarity of failure. For instance, if $n = 1000$ and $c = 4$, we may conclude that the chance that the running time of the algorithm A exceeding the threshold value is one in zillion.

13.2.2 Basics of Probability Theory

We assume that the reader is familiar with basic notions such as *sample space, event* and basic *axioms of probability*. We denote as $Pr(E)$ the probability of the event E. Several results follow immediately from the basic axioms, and some of them are listed in Lemma 13.1.

LEMMA 13.1 The following laws of probability must hold:

1. $Pr(\phi) = 0$
2. $Pr(E^c) = 1 - Pr(E)$
3. $Pr(E_1) \leq Pr(E_2)$ if $E_1 \subseteq E_2$
4. $Pr(E_1 \cup E_2) = Pr(E_1) + Pr(E_2) - Pr(E_1 \cap E_2) \leq Pr(E_1) + Pr(E_2)$

Extending item 4 in Lemma 13.1 to countable unions yields the property known as *sub additivity*. Also known as *Boole's Inequality*, it is stated in Theorem 13.1.

THEOREM 13.1 *[Boole's Inequality]* $Pr(\cup_{i=1}^{\infty} E_i) \leq \sum_{i=1}^{\infty} Pr(E_i)$

A probability distribution is said to be *discrete* if the sample space S is finite or countable. If $E = \{e_1, e_2, ..., e_k\}$ is an event, $Pr(E) = \sum_{i=1}^{k} Pr(\{e_i\})$ because all elementary events are mutually exclusive. If $|S| = n$ and $Pr(\{e\}) = \frac{1}{n}$ for every elementary event e in S, we call the distribution a *uniform distribution* of S. In this case,

$$\begin{aligned} Pr(E) &= \sum_{e \in E} Pr(e) \\ &= \sum_{e \in E} \frac{1}{n} \\ &= |E|/|S| \end{aligned}$$

which agrees with our intuitive and a well-known definition that probability is the ratio of the favorable number of cases to the total number of cases, when all the elementary events are equally likely to occur.

13.2.3 Conditional Probability

In several situations, the occurrence of an event may change the uncertainties in the occurrence of other events. For instance, insurance companies charge higher rates to various categories of drivers, such as those who have been involved in traffic accidents, because the probabilities of these drivers filing a claim is altered based on these additional factors.

DEFINITION 13.3 [Conditional Probability] The *conditional probability* of an event E_1 given that another event E_2 has occurred is defined by $Pr(E_1/E_2)$ ("$Pr(E_1/E_2)$" is read as "the probability of E_1 given E_2.").

LEMMA 13.2 $Pr(E_1/E_2) = \frac{Pr(E_1 \cap E_2)}{Pr(E_2)}$, provided $Pr(E_2) \neq 0$.

Lemma 13.2 shows that the conditional probability of two events is easy to compute. When two or more events do not influence each other, they are said to be independent. There are several notions of independence when more than two events are involved. Formally,

DEFINITION 13.4 [Independence of two events] Two events are *independent* if $Pr(E_1 \cap E_2) = Pr(E_1)Pr(E_2)$, or equivalently, $Pr(E_1/E_2) = Pr(E_1)$.

DEFINITION 13.5 [Pairwise independence] Events $E_1, E_2, \ldots E_k$ are said to be *pairwise independent* if $Pr(E_i \cap E_j) = Pr(E_i)Pr(E_j)$, $1 \leq i \neq j \leq n$.

Given a partition $S_1, \ldots, S_k$ of the sample space S, the probability of an event E may be expressed in terms of mutually exclusive events by using conditional probabilities. This is known as the *law of total probability in the conditional form*.

LEMMA 13.3 [Law of total probability in the conditional form] For any partition $S_1, ..., S_k$ of the sample space S, $Pr(E) = \sum_{i=1}^{k} Pr(E/S_i)\ Pr(S_i)$.

The law of total probability in the conditional form is an extremely useful tool for calculating the probabilities of events. In general, to calculate the probability of a complex event E, we may attempt to find a partition $S_1, S_2, \ldots, S_k$ of S such that both $Pr(E/S_i)$ and $Pr(S_i)$ are easy to calculate and then apply Lemma 13.3. Another important tool is *Bayes' Rule*.

THEOREM 13.2 *[Bayes' Rule] For events with non-zero probabilities,*

1. $Pr(E_1/E_2) = \frac{Pr(E_2/E_1)Pr(E_1)}{Pr(E_2)}$
2. *If* $S_1, S_2, ..., S_k$ *is a partition,* $Pr(S_i/E) = \frac{Pr(E/S_i)Pr(S_i)}{\sum_{j=1} Pr(E/S_j)Pr(S_j)}$

Proof Part (1) is immediate by applying the definition of conditional probability; Part (2) is immediate from Lemma 13.3.

Random Variables and Expectation

Most of the random phenomena are so complex that it is very difficult to obtain detailed information about the outcome. Thus, we typically study one or two numerical parameters that we associate with the outcomes. In other words, we focus our attention on certain real-valued functions defined on the sample space.

DEFINITION 13.6 A *random variable* is a function from a sample space into the set of real numbers. For a random variable X, $R(X)$ denotes the *range* of the function X.

Having defined a random variable over a sample space, an event of interest may be studied through the values taken by the random variables on the outcomes belonging to the event. In order to facilitate such a study, we supplement the definition of a random variable by specifying how the probability is assigned to (or distributed over) the values that the random variable may assume. Although a rigorous study of random variables will require a more subtle definition, we restrict our attention to the following simpler definitions that are sufficient for our purposes.

A random variable X is a *discrete random variable* if its range $R(X)$ is a finite or countable set (of real numbers). This immediately implies that any random variable that is defined over a finite or countable sample space is necessarily discrete. However, discrete random variables may also be defined on uncountable sample spaces. For a random variable X, we define its *probability mass function (pmf)* as follows:

DEFINITION 13.7 [Probability mass function] For a random variable X, the *probability mass function* $p(x)$ is defined as $p(x) = Pr(X = x)$, $\forall x \in R(X)$.

The probability mass function is also known as the *probability density function*. Certain trivial properties are immediate, and are given in Lemma 13.4.

LEMMA 13.4 The probability mass function $p(x)$ must satisfy

1. $p(x) \geq 0$, $\forall x \in R(X)$
2. $\sum_{x \in R(X)} p(x) = 1$

Let X be a discrete random variable with probability mass function $p(x)$ and range $R(X)$. The *expectation* of X (also known as the *expected value* or *mean* of X) is its average value. Formally,

DEFINITION 13.8 [Expected value of a discrete random variable] The *expected value* of a discrete random variable X with probability mass function $p(x)$ is given by $E(X) = \mu_X = \sum_{x \in R(X)} xp(x)$.

LEMMA 13.5 The expected value has the following properties:

1. $E(cX) = cE(X)$ if c is a constant

2. (Linearity of expectation) $E(X+Y) = E(X) + E(Y)$, provided the expectations of X and Y exist

Finally, a useful way of computing the expected value is given by Theorem 13.3.

THEOREM 13.3 *If $R(X) = \{0, 1, 2, ...\}$, then $E(X) = \sum_{i=1}^{\infty} Pr(X \geq i)$.*

Proof

$$\begin{aligned} E(X) &= \sum_{i=0}^{\infty} iPr(X = i) \\ &= \sum_{i=0}^{\infty} i(Pr(X \geq i) - Pr(X \geq i+1)) \\ &= \sum_{i=1}^{\infty} Pr(X \geq i) \end{aligned}$$

13.2.4 Some Basic Distributions

Bernoulli Distribution

We have seen that a coin flip is an example of a random experiment and it has two possible outcomes, called *success* and *failure*. Assume that success occurs with probability p and that failure occurs with probability $q = 1 - p$. Such a coin is called p-biased coin. A coin flip is also known as *Bernoulli Trial*, in honor of the mathematician who investigated extensively the distributions that arise in a sequence of coin flips.

DEFINITION 13.9 A random variable X with range $R(X) = \{0, 1\}$ and probability mass function $Pr(X = 1) = p$, $Pr(X = 0) = 1 - p$ is said to follow the Bernoulli Distribution. We also say that X is a Bernoulli random variable with parameter p.

Binomial Distribution

Let $\left(\begin{array}{c} n \\ k \end{array} \right)$ denote the number of k-combinations of elements chosen from a set of n elements. Recall that $\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!}$ and $\left(\begin{array}{c} n \\ 0 \end{array} \right) = 1$ since $0! = 1$. $\left(\begin{array}{c} n \\ k \end{array} \right)$ denotes the *binomial coefficients* because they arise in the expansion of $(a+b)^n$.

Define the random variable X to be the number of successes in n flips of a p-biased coin. The variable X satisfies the *binomial distribution*. Specifically,

DEFINITION 13.10 [Binomial distribution] A random variable with range $R(X) = \{0, 1, 2, \ldots, n\}$ and probability mass function

$$Pr(X = k) = b(k, n, p) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k q^{n-k}, \quad for \quad k = 0, 1, \ldots, n$$

satisfies the *binomial distribution*. The random variable X is called a binomial random variable with parameters n and p.

THEOREM 13.4 *For a binomial random variable X, with parameters n and p, $E(X) = np$ and $Var(X) = npq$.*

Geometric Distribution

Let X be a random variable X denoting the number of times we toss a p-biased coin until we get a success. Then, X satisfies the *geometric distribution*. Specifically,

DEFINITION 13.11 [Geometric distribution] A random variable with range $R(X) = \{1, 2, \ldots, \infty\}$ and probability mass function $Pr(X = k) = q^{k-1}p$, for $k = 1, 2, \ldots, \infty$ satisfies the *geometric distribution*. We also say that X is a geometric random variable with parameter p.

The probability mass function is based on $k-1$ failures followed by a success in a sequence of k independent trials. The mean and variance of a geometric distribution are easy to compute.

THEOREM 13.5 *For a geometrically distributed random variable X, $E(X) = \frac{1}{p}$ and $Var(X) = \frac{q}{p^2}$.*

Negative Binomial distribution

Fix an integer n and define a random variable X denoting the number of flips of a p-biased coin to obtain n successes. The variable X satisfies a negative binomial distribution. Specifically,

DEFINITION 13.12 A random variable X with $R(X) = \{0, 1, 2, ...\}$ and probability mass function defined by

$$\begin{aligned} Pr(X = k) &= \binom{k-1}{n-1} p^n q^{k-n} \quad if\ k \geq n \\ &= 0 \quad if\ 0 \leq k < n \end{aligned} \tag{13.1}$$

is said to be a *negative binomial random variable* with parameters n and p.

Equation (13.1) follows because, in order for the n^{th} success to occur in the k^{th} flip there should be $n-1$ successes in the first $k-1$ flips and the k^{th} flip should also result in a success.

DEFINITION 13.13 Given n identically distributed independent random variables $X_1, X_2, \ldots, X_n$, the sum

$$S_n = X_1 + X_2 + \cdots + X_n$$

defines a new random variable. If n is a finite, fixed constant then S_n is known as the *deterministic sum* of n random variables.

On the other hand, if n itself is a random variable, S_n is called a *random sum*.

THEOREM 13.6 *Let $X = X_1 + X_2 + \cdots + X_n$ be a deterministic sum of n identical independent random variables. Then*

1. *If X_i is a Bernoulli random variable with parameter p then X is a binomial random variable with parameters n and p.*
2. *If X_i is a geometric random variable with parameter p, then X is a negative binomial with parameters n and p.*
3. *If X_i is a (negative) binomial random variable with parameters r and p then X is a (negative) binomial random variable with parameters nr and p.*

Deterministic sums and random sums may have entirely different characteristics as the following theorem shows.

THEOREM 13.7 *Let $X = X_1 + \cdots + X_N$ be a random sum of N geometric random variables with parameter p. Let N be a geometric random variable with parameter α. Then X is a geometric random variable with parameter αp.*

13.2.5 Tail Estimates

Recall that the running time of a *Las Vegas* type randomized algorithm is a random variable and thus we are interested in the probability of the running time exceeding a certain threshold value.

Typically we would like this probability to be very small so that the threshold value may be taken as the figure of merit with high degree of confidence. Thus we often compute or estimate quantities of the form $Pr(X \geq k)$ or $Pr(X \leq k)$ during the analysis of randomized algorithms. Estimates for the quantities of the form $Pr(X \geq k)$ are known as *tail estimates*. The next two theorems state some very useful tail estimates derived by Chernoff. These bounds are popularly known as *Chernoff bounds*. For simple and elegant proofs of these and other related bounds you may refer [1].

THEOREM 13.8 *Let X be a sum of n independent random variables X_i with $R(X_i) \subseteq [0,1]$. Let $E(X) = \mu$. Then,*

$$Pr(X \geq k) \leq \left(\frac{\mu}{k}\right)^k \left(\frac{n-\mu}{n-k}\right)^{n-k} \quad \text{for } k > \mu \tag{13.2}$$

$$\leq \left(\frac{\mu}{k}\right)^k e^{k-\mu} \quad \text{for } k > \mu \tag{13.3}$$

$$Pr(X \geq (1+\epsilon)\mu) \leq \left[\frac{e^\epsilon}{(1+\epsilon)^{(1+\epsilon)}}\right]^\mu \quad \text{for } \epsilon \geq 0 \tag{13.4}$$

THEOREM 13.9 *Let X be a sum of n independent random variables X_i with $R(X_i) \subseteq [0,1]$. Let $E(X) = \mu$. Then,*

$$Pr(X \leq k) \leq \left(\frac{\mu}{k}\right)^k \left(\frac{n-\mu}{n-k}\right)^{n-k} \quad k < \mu \tag{13.5}$$

$$\leq \left(\frac{\mu}{k}\right)^k e^{k-\mu} \quad k < \mu \tag{13.6}$$

$$Pr(X \leq (1-\epsilon)\mu) \leq e^{-\frac{\mu\epsilon^2}{2}}, \quad \text{for } \epsilon \in (0,1) \tag{13.7}$$

Recall that a deterministic sum of several geometric variables results in a negative binomial random variable. Hence, intuitively, we may note that only the upper tail is meaningful for this distribution. The following well-known result relates the upper tail value of a negative binomial distribution to a lower tail value of a suitably defined binomial distribution. Hence all the results derived for lower tail estimates of the binomial distribution can be used to derive upper tail estimates for negative binomial distribution. This is a very important result because finding bounds for the right tail of a negative binomial distribution directly from its definition is very difficult.

THEOREM 13.10 *Let X be a negative binomial random variable with parameters r and p. Then, $Pr(X > n) = Pr(Y < r)$ where Y is a binomial random variable with parameters n and p.*

13.3 Skip Lists

Linked list is the simplest of all dynamic data structures implementing a *Dictionary*. However, the complexity of *Search* operation is $O(n)$ in a *Linked list*. Even the *Insert* and *Delete* operations require $O(n)$ time if we do not specify the exact position of the item in the list. *Skip List* is a novel structure, where using randomness, we construct a number of progressively smaller lists and maintain this collection in a clever way to provide a data structure that is competitive to balanced tree structures. The main advantage offered by skip list is that the codes implementing the dictionary operations are very simple and resemble list operations. No complicated structural transformations such as *rotations* are done and yet the expected time complexity of *Dictionary* operations on *Skip Lists* are quite comparable to that of AVL trees or splay trees. *Skip Lists* are introduced by Pugh [6].

Throughout this section let $S = \{k_1, k_2, \ldots, k_n\}$ be the set of keys and assume that $k_1 < k_2 < \ldots < k_n$.

DEFINITION 13.14 Let $S_0, S_1, S_2, \ldots, S_r$ be a collection of sets satisfying

$$S = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots \supset S_r = \phi$$

Then, we say that the collection $S_0, S_1, S_2, \ldots, S_r$ defines a *leveling with r levels on S*. The keys in S_i are said to be in level i, $0 \leq i \leq r$. The set S_0 is called the base level for the leveling scheme. Notice that there is exactly one empty level, namely S_r. The level number $l(k)$ of a key $k \in S$ is defined by

$$l(k) = max\{i \mid k \in L_i\}.$$

In other words, $k \in S_0, S_1, S_2, \ldots, S_{l(k)}$ but $k \notin S_{l(k)+1} \cdots S_r$.

For an efficient implementation of the dictionary, instead of working with the current set S of keys, we would rather work with a leveling of S. The items of S_i will be put in the *increasing order* in a linked list denoted by L_i. We attach the special keys $-\infty$ at the beginning and $+\infty$ at the end of each list L_i as sentinels. In practice, $-\infty$ is a key value that is smaller than any key we consider in the application and $+\infty$ denotes a key value larger than all the possible keys. A leveling of S is implemented by maintaining all the lists

$L_0, L_1, L_2, \ldots, L_r$ with some more additional links as shown in Figure 13.1. Specifically, the box containing a key k in L_i will have a pointer to the box containing k in L_{i-1}. We call such pointers *descent pointers*. The links connecting items of the same list are called *horizontal pointers*. Let B be a pointer to a box in the skip list. We use the notations $Hnext[B]$, and $Dnext[B]$, for the horizontal and descent pointers of the box pointed by B respectively. The notation $key[B]$ is used to denote the key stored in the box pointed by B. The name of the skip list is nothing but a pointer to the box containing $-\infty$ in the rth level as shown in the figure. From the Figure 13.1 it is clear that L_i has horizontal pointers that skip over several intermediate elements of L_{i-1}. That is why this data structure is called the *Skip List*.

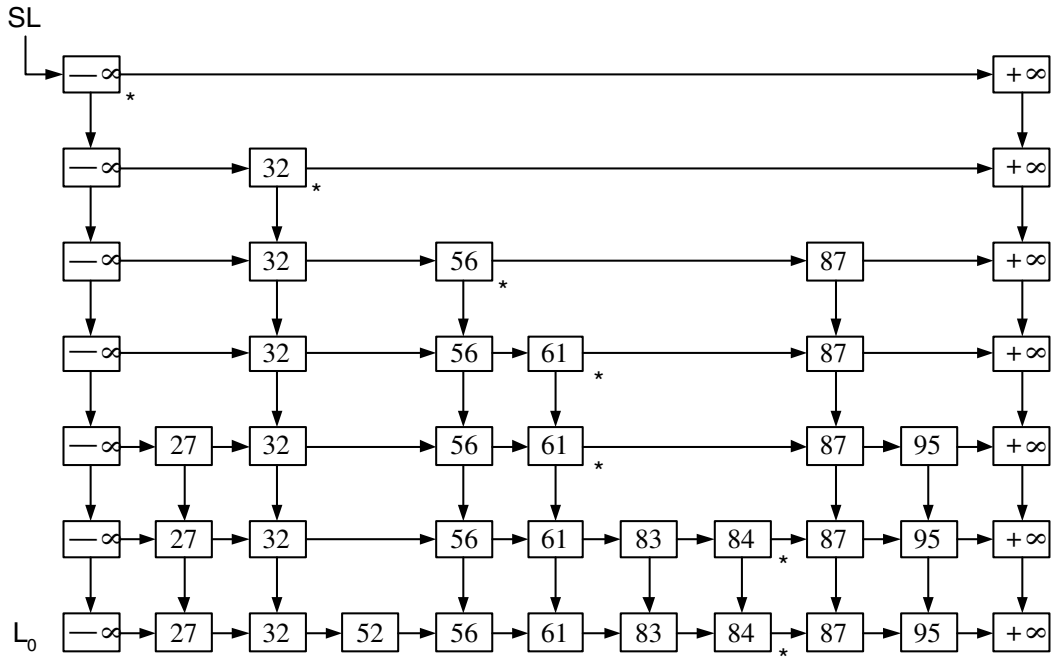


FIGURE 13.1: A Skip List. The starred nodes are marked by Mark(86,SL).

How do we arrive at a leveling for a given set? We may construct S_{i+1} from S_i in a systematic, deterministic way. While this may not pose any problem for a static search problem, it will be extremely cumbersome to implement the dynamic dictionary operations. This is where randomness is helpful. To get S_{i+1} from S_i, we do the following. For each element k of S_i toss a coin and include k in S_{i+1} iff we get *success* in the coin toss. If the coin is p-biased, we expect $\mid S_{i+1} \mid$ to be roughly equal to $p \mid S_i \mid$. Starting from S, we may repeatedly obtain sets corresponding to successive levels. Since the coin tosses are independent, there is another useful, and equivalent way to look at our construction. For each key k in S, keep tossing a coin until we get a *failure*. Let h be the number of successes before the first failure. Then, we include k in h further levels treating $S = S_0$ as the base level. In other words, we include k in the sets $S_1, S_2, \ldots, S_h$. This suggests us to define a random variable Z_i for each key $k_i \in S_0$ to be the number of times we toss a p-biased coin

before we obtain a *failure*. Since Z_i denotes the number of additional copies of k_i in the skip list, the value $maximum\{Z_i : 1 \leq i \leq n\}$ defines the highest nonempty level. Hence, we have,

$$r = 1 + maximum\{Z_i : 1 \leq i \leq n\} \tag{13.8}$$

$$\mid SL \mid = n + Z_1 + Z_2 + \cdots + Z_n + 2r + 2 \tag{13.9}$$

where r is the number of levels and $\mid SL \mid$ is the number of boxes or the space complexity measure of the *Skip List*. In the expression for $\mid SL \mid$ we have added n to count the keys in the base level and $2r + 2$ counts the sentinel boxes containing $+\infty$ and $-\infty$ in each level.

13.4 Structural Properties of Skip Lists

13.4.1 Number of Levels in Skip List

Recall that $r = 1 + max_i\{Z_i\}$. Notice that $Z_i \geq k$ iff the coin tossed for k_i gets a run of at least k successes right from the beginning and this happens with probability p^k. Since $r \geq k$ iff at least one of $Z_i \geq k - 1$, we easily get the following fact from Boole's inequality

$$Pr(r \geq k) \leq np^{k-1}$$

Choosing $k = 4\log_{1/p} n + 1$, we immediately obtain a high confidence bound for the number of levels. In fact,

$$\begin{aligned} Pr(r \geq 4\log n + 1) &\leq nn^{-4} \\ &= \frac{1}{n^3} \end{aligned} \tag{13.10}$$

We obtain an estimation for the expected value of r, using the formula stated in theorem (13.3) as follows:

$$\begin{aligned} E(r) &= \sum_{i=1}^{\infty} Pr(r \geq i) \\ &= \sum_{i=1}^{4\log n} Pr(r \geq i) + \sum_{i > 4\log n} Pr(r \geq i) \\ &\leq \sum_{i=1}^{4\log n} 1 + \sum_{i > 4\log n}^{\infty} np^{i-1} \\ &= 4\log n + np^{4\log n}(1 + p + p^2 + \cdots) \\ &= 4\log n + n \cdot \frac{1}{n^4} \cdot (1-p)^{-1} \text{ if base of the log is 1/p} \\ &\leq 4\log n + 1 \text{ for sufficiently large } n \end{aligned}$$

Thus $E(r) = O(\log n)$
Hence,

THEOREM 13.11 *The expected number of levels in a skip list of n elements is $O(\log n)$. In fact, the number is $O(\log n)$ with high probability.*

13.4.2 Space Complexity

Recall that the space complexity, $\mid SL \mid$ is given by

$$\mid SL \mid = Z_1 + Z_2 + \cdots + Z_n + n + 2r + 2.$$

As we know that $r = O(\log n)$ with high probability, let us focus on the sum

$$Z = Z_1 + Z_2 + \cdots + Z_n.$$

Since Z_i is a geometric random variable with parameter p, Z is a negative binomial random variable by theorem 13.6.

Thus $E(Z) = \frac{n}{p} = O(n)$.

We can in fact show that Z is $O(n)$ with very high probability.

Now, from theorem (13.10) $Pr(Z > 4n) = Pr(X < n)$ where X is a binomial distribution with parameters $4n$ and p. We now assume that $p = 1/2$ just for the sake of simplicity in arithmetic.

In the first Chernoff bound mentioned in theorem (13.9), replacing n, μ and k respectively with $4n$, $2n$ and n, we obtain,

$$\begin{aligned} Pr(X < n) &\leq \left(\frac{2n}{n}\right)^n \left(\frac{2n}{3n}\right)^{3n} \\ &= \left(2 \cdot \frac{2^3}{3^3}\right)^n \\ &= \left(\frac{16}{27}\right)^n \\ &= \left(\frac{27}{16}\right)^{-n} \end{aligned}$$

This implies that $4n$ is in fact a very high confidence bound for Z. Since $\mid SL \mid = Z+n+2r+2$, we easily conclude that

THEOREM 13.12 *The space complexity of a skip list for a set of size n is $O(n)$ with very high probability.*

13.5 Dictionary Operations

We shall use a simple procedure called *Mark* in all the dictionary operations. The procedure *Mark* takes an arbitrary value x as input and marks in each level the box containing the largest key that is less than x. This property implies that insertion, deletion or search should all be done next to the marked boxes. Let $M_i(x)$ denote the box in the i^{th} level that is marked by the procedure call *Mark*(x, SL). Recall the convention used in the linked structure that name of a box in a linked list is nothing but a pointer to that box. The keys in the marked boxes $M_i(x)$ satisfy the following condition :

$$key[M_i(x)] < x \leq key[Hnext[M_i(x)]] \quad \text{for all} \;\; 0 \leq i \leq r. \tag{13.11}$$

The procedure *Mark* begins its computation at the box containing $-\infty$ in level r. At any current level, we traverse along the level using horizontal pointers until we find that the

next box is containing a key larger or equal to x. When this happens, we mark the current box and use the *descent pointer* to reach the next lower level and work at this level in the same way. The procedure stops after marking a box in level 0. See Figure 13.1 for the nodes marked by the call *Mark(86,SL)*.

```
Algorithm Mark(x,SL)
-- x is an arbitrary value.
-- r is the number of levels in the skip list SL.
1. Temp  = SL.
2. For i = r down to 0 do
     While (key[Hnext[Temp]] < x)
            Temp = Hnext[Temp];
     Mark the box pointed by Temp;
     Temp = Dnext[Temp];
3. End.
```

We shall now outline how to use marked nodes for *Dictionary* operations. Let $M_0(x)$ be the box in level 0 marked by *Mark(x)*. It is clear that the box next to $M_0(x)$ will have x iff $x \in S$. Hence our algorithm for *Search* is rather straight forward.

To insert an item in the *Skip List*, we begin by marking the *Skip List* with respect to the value x to be inserted. We assume that x is not already in the *Skip List*. Once the marking is done, inserting x is very easy. Obviously x is to be inserted next to the marked boxes. But in how many levels we insert x? We determine the number of levels by tossing a coin on behalf of x until we get a *failure*. If we get h successes, we would want x to be inserted in all levels from 0 to h. If $h \geq r$, we simply insert x at all the existing levels and create a new level consisting of only $-\infty$ and $+\infty$ that corresponds to the empty set. The insertion will be done starting from the base level. However, the marked boxes are identified starting from the highest level. This means, the *Insert* procedure needs the marked boxes in the reverse order of its generation. An obvious strategy is to push the marked boxes in a stack as and when they are marked in the *Mark* procedure. We may pop from the stack as many marked boxes as needed for insertion and then insert x next to each of the popped boxes.

The deletion is even simpler. To delete a key x from S, simply mark the *Skip List* with respect to x. Note that if x is in L_i, then it will be found next to the marked box in L_i. Hence, we can use the *horizontal pointers* in the marked boxes to delete x from each level where x is present. If the deletion creates one or more empty levels, we ignore all of them and retain only one level corresponding to the empty set. In other words, the number of levels in the *Skip List* is reduced in this case. In fact, we may delete an item "on the fly" during the execution of the *Mark* procedure itself. As the details are simple we omit pseudo codes for these operations.

13.6 Analysis of Dictionary Operations

It is easy to see that the cost of *Search*, *Insert* and *Delete* operations are dominated by the cost of *Mark* procedure. Hence we shall analyze only the *Mark* procedure. The *Mark* procedure starts at the 'r'th level and proceeds like a downward walk on a staircase and ends at level 0. The complexity of *Mark* procedure is clearly proportional to the number of edges it traversed. It is advantageous to view the same path as if it is built from level 0 to level r. In other words, we analyze the building of the path in a direction opposite to the direction in which it is actually built by the procedure. Such an analysis is known as *backward analysis*.

Henceforth let P denote the path from level 0 to level r traversed by *Mark(x)* for the given fixed x. The path P will have several vertical edges and horizontal edges. (Note that at every box either P moves vertically above or moves horizontally to the left). Clearly, P has r vertical edges. To estimate number of horizontal edges, we need the following lemmas.

LEMMA 13.6 Let the box b containing k at level i be marked by Mark(x). Let a box w containing k be present at level $i+1$. Then, w is also marked.

Proof Since b is marked, from (13.11), we get that there is no value between k and x in level i. This fact holds good for L_{i+1} too because $S_{i+1} \subseteq S_i$. Hence the lemma.

LEMMA 13.7 Let the box b containing k at level i be marked by Mark(x). Let $k \notin L_{i+1}$. Let u be the first box to the left of b in L_i having a "vertical neighbor" w. Then w is marked.

Proof Let $w.key = u.key = y$. Since b is marked, k satisfies condition (13.11). Since u is the first node in the left of b having a vertical neighbor, none of the keys with values in between y and x will be in L_{i+1}. Also, $k \notin L_{i+1}$ according to our assumption in the lemma. Thus y is the element in L_{i+1} that is just less than x. That is, y satisfies the condition (13.11) at level $i+1$. Hence the w at level $i+1$ will be marked.

Lemmas (13.6) and (13.7) characterize the segment of P between two successive marked boxes. This allows us to give an incremental description of P in the following way.

P starts from the marked box at level 0. It proceeds vertically as far as possible (lemma 13.6) and when it cannot proceed vertically it proceeds horizontally. At the "first" opportunity to move vertically, it does so (lemma 13.7), and continues its vertical journey. Since for any box a vertical neighbor exists only with a probability p, we see that P proceeds from the current box vertically with probability p and horizontally with probability $(1-p)$.

Hence, the number of horizontal edges of P in level i is a geometric random variable, say, Y_i, with parameter $(1-p)$. Since the number of vertical edges in P is exactly r, we conclude,

THEOREM 13.13 *The number of edges traversed by Mark(x) for any fixed x is given by $\mid P \mid = r + (Y_0 + Y_1 + Y_2 + \cdots + Y_{r-1})$ where Y_i is a geometric random variable with parameters $1-p$ and r is the random variable denoting the number of levels in the Skip List.*

Our next mission is to obtain a high confidence bound for the size of P. As we have already derived high confidence bound for r, let focus on the sum $H_r = Y_0 + Y_1 + \cdots + Y_{r-1}$ for a while. Since r is a random variable H_r is not a deterministic sum of random variables but a *random sum* of random variables.

Hence we cannot directly apply theorem (13.6) and the bounds for negative binomial distributions.

Let X be the event of the random variable r taking a value less than or equal to $4 \log n$. Note that $P(\overline{X}) < \frac{1}{n^3}$ by (13.10).

From the law of total probability, Boole's inequality and equation (13.10) we get,

$$\begin{aligned} Pr(H_r > 16\log n) &= Pr([H_r > 16\log n] \cap X) + Pr([H_r > 16\log n] \cap \overline{X}) \\ &= Pr([H_r > 16\log n] \cap r \leq 4\log n) + Pr([H_r > 16\log n] \cap \overline{X}) \\ &\leq \sum_{k=0}^{4\log n} Pr(H_k > 16\log n) + Pr(\overline{X}) \\ &\leq (1 + 4\log n) Pr(H_{4\log n} > 16\log n) + \frac{1}{n^3} \end{aligned}$$

Now $Pr(H_{4\log n} > 16\log n)$ can be computed in a manner identical to the one we carried out in the space complexity analysis. Notice that $H_{4\log n}$ is a deterministic sum of geometric random variables. Hence we can apply theorem (13.10) and theorem (13.9) to derive a high confidence bound. Specifically, by theorem (13.10),

$$Pr(H_{4\log n} > 16\log n) = Pr(X < 4\log n),$$

where X is a binomial random variable with parameters $16\log n$ and p. Choosing $p = 1/2$ allows us to set $\mu = 8\log n$, $k = 4\log n$ and replace n by $16\log n$ in the first inequality of theorem (13.9). Putting all these together we obtain,

$$\begin{aligned} Pr(H_{4\log n} > 16\log n) &= Pr(X < 4\log n) \\ &\leq \left(\frac{8\log n}{4\log n}\right)^{4\log n} \left(\frac{8\log n}{12\log n}\right)^{12\log n} \\ &= \left(2 \cdot \frac{2^3}{3^3}\right)^{4\log n} \\ &= \left(\frac{16}{27}\right)^{4\log n} \\ &< \left(\frac{1}{8}\right)^{\log n} \\ &= \frac{1}{n^3} \end{aligned}$$

Therefore

$$\begin{aligned} Pr(H_r > 16\log n) &< (1 + 4\log n)\left(\frac{1}{n^3}\right) + \frac{1}{n^3} \\ &< \frac{1}{n^2} \quad if \quad n \geq 32 \end{aligned}$$

This completes the derivation of a high confidence bound for H_r. From this, we can easily obtain a bound for expected value of H_r. We use theorem (13.3) and write the expression for $E(H_r)$ as

$$E(H_r) = \sum_{i=1}^{16\log n} Pr(H_r \geq i) + \sum_{i > 16\log n} Pr(H_r \geq i).$$

The first sum is bounded above by $16\log n$ as each probability value is less than 1 and by the high confidence bound that we have established just now, we see that the second sum is dominated by $\sum_{i=1}^{\infty} 1/i^2$ which is a constant. Thus we obtain,

$$E(H_r) \leq 16 \log n + c = O(\log n).$$

Since $P = r + H_r$ we easily get that $E(\mid P \mid) = O(\log n)$ and $\mid P \mid = O(\log n)$ with probability greater than $1 - O(\frac{1}{n^2})$. Observe that we have analyzed *Mark(x)* for a given fixed x. To show that the high confidence bound for any x, we need to proceed little further. Note that there are only $n + 1$ distinct paths possible with respect to the set $\{-\infty = k_0, k_1, ..., k_n, k_{n+1} = +\infty\}$, each corresponding to the x lying in the internal $[k_i, k_{i+1}), i = 0, 1, ..., n$.

Therefore, for any x, *Mark(x)* walks along a path P satisfying $E(\mid P \mid) = O(\log n)$ and $\mid P \mid = O(\log n)$ with probability greater than $1 - O(\frac{1}{n})$.

Summarizing,

THEOREM 13.14 *The Dictionary operations Insert, Delete, and Search take $O(\log n)$ expected time when implemented using Skip Lists. Moreover, the running time for Dictionary operations in a Skip List is $O(\log n)$ with high probability.*

13.7 Randomized Binary Search Trees

A *Binary Search Tree* (BST) for a set S of keys is a binary tree satisfying the following properties.

(a) Each node has exactly one key of S. We use the notation $v.key$ to denote the key stored at node v.

(b) For all node v and for all nodes u in the left subtree of v and for all nodes w in the right subtree of v, the keys stored in them satisfy the so called *search tree property*:

$$u.key < v.key < w.key$$

The complexity of the dictionary operations are bounded by the height of the binary search tree. Hence, ways and means were explored for controlling the height of the tree during the course of execution. Several clever balancing schemes were discovered with varying degrees of complexities. In general, the implementation of balancing schemes are tedious and we may have to perform a number of rotations which are complicated operations. Another line of research explored the potential of possible randomness in the input data. The idea was to completely avoid balancing schemes and hope to have '*short*' trees due to randomness in input. When only random insertion are done, we obtain so called *Randomly Built Binary Tree* (RBBT). RBBTs have been shown to have $O(\log n)$ expected height.

What is the meaning of random insertion? Suppose we have already inserted the values $a_1, a_2, a_3, \cdots, a_{k-1}$. These values, when considered in sorted order, define k intervals on the real line and the new value to be inserted, say x, is equally likely to be in any of the k intervals.

The first drawback of RBBT is that this assumption may not be valid in practice and when this assumption is not satisfied, the resulting tree structure could be highly skewed and the complexity of search as well as insertion could be as high as $O(n)$. The second major drawback is when deletion is also done on these structures, there is a tremendous degradation in the performance. There is no theoretical results available and extensive

empirical studies show that the height of an RBBT could grow to $O(\sqrt{n})$ when we have arbitrary mix of insertion and deletion, even if the randomness assumption is satisfied for inserting elements. Thus, we did not have a satisfactory solution for nearly three decades.

In short, the randomness was not preserved by the deletion operation and randomness preserving binary tree structures for the dictionary operations was one of the outstanding open problems, until an elegant affirmative answer is provided by Martinez and Roura in their landmark paper [3].

In this section, we shall briefly discuss about structure proposed by Martinez and Roura.

DEFINITION 13.15 [Randomized Binary Search Trees] Let T be a binary search tree of size n. If $n = 0$, then $T = NULL$ and it is a random binary search tree. If $n > 0$, T is a random binary search tree iff both its left subtree L and right subtree R are independent random binary search trees and

$$\Pr\{Size(L) = i | Size(T) = n\} = \frac{1}{n},\ 0 \leq i \leq n.$$

The above definition implies that every key has the same probability of $\frac{1}{n}$ for becoming the root of the tree. It is easy to prove that the expected height of a RBST with n nodes is $O(\log n)$. The RBSTs possess a number of interesting structural properties and the classic book by Mahmoud [2] discusses them in detail. In view of the above fact, it is enough if we prove that when insert or delete operation is performed on a RBST, the resulting tree is also an RBST.

13.7.1 Insertion in RBST

When a key x is inserted in a tree T of size n, we obtain a tree T' of size $n + 1$. For T', as we observed earlier, x should be in the root with probability $\frac{1}{n+1}$. This is our starting point.

```
Algorithm Insert(x, T)
- L is the left subtree of the root
- R is the right subtree of the root
1. n = size(T);
2. r = random(0, n);
3. If (r = n) then
      Insert_at_root(x, T);
4. If (x < key at root of T) then
      Insert(x, L);
   Else
      Insert(x, R);
```

To insert x as a root of the resulting tree, we first split the current tree into two trees labeled $T_<$ and $T_>$, where $T_<$ contains all the keys in T that are smaller than x and $T_>$ contains all the keys in T that are larger than x. The output tree T' is constructed by placing x at the root and attaching $T_<$ as its left subtree and $T_>$ as its right subtree. The algorithm for splitting a tree T into $T_<$ and $T_>$ with respect to a value x is similar to the partitioning algorithm done in *quicksort*. Specifically, the algorithm *split(x,T)* works as

follows. If T is empty , nothing needs to be done; both $T_<$ and $T_>$ are empty. When T is non-empty we compare x with Root(T).key. If $x <$ Root(T).key, then root of T as well as the right subtree of T belong to $T_>$. To compute $T_<$ and the remaining part of $T_>$ we recursively call *split*(x, L), where L is the left subtree for the root of T. If $x >$ Root(T).key, $T_<$ is built first and recursion proceeds with *split*(x, R). The details are left as easy exercise.

We shall first prove that $T_<$ and $T_>$ are independent *Random Binary Search Trees*. Formally,

THEOREM 13.15 *Let $T_<$ and $T_>$ be the BSTs produced by split(x, T). If T is a random BST containing the set of keys S, then $T_<$ and $T_>$ are RBBTs containing the keys $S_< = \{y \in S \mid y < x\}$ and $S_> = \{y \in S \mid y > x\}$, respectively.*

Proof Let size(T) $= n > 0$, $x >$ Root(T).key, we will show that for any $z \in S_<$, the probability that z is the root of $T_<$ is $1/m$ where $m =$ size($T_<$). In fact,

$$\begin{aligned} & \Pr(\ z \text{ is root of } T_< \mid \text{root of } T \text{ is less than } x) \\ = \ & \frac{\Pr(\ z \text{ is root of } T_< \text{ and root of } T \text{ is less than } x)}{\Pr(\text{root of } T \text{ is less than } x)} \\ = \ & \frac{1/n}{m/n} = \frac{1}{m}. \end{aligned}$$

The independence of $T_<$ and $T_>$ follows from the independence of L and R of T and by induction.

We are now ready to prove that randomness is preserved by insertion. Specifically,

THEOREM 13.16 *Let T be a RBST for the set of keys S and $x \notin S$, and assume that insert(s, T) produces a tree, say T' for $S \cup \{x\}$. Then, T' is a RBST.*

Proof A key $y \in S$ will be at the root of T' iff

1) y is at root of T.
2) x is not inserted at the root of T'

As 1) and 2) are independent events with probability $\frac{1}{n}$ and $\frac{n}{n+1}$, respectively, it follows that Prob(y is at root of T') $= \frac{1}{n} \cdot \frac{n}{n+1} = \frac{1}{n+1}$. The key x can be at the root of T' only when *insert*(x, T) invokes *insert-at-root*(x, T) and this happens with probability $\frac{1}{n+1}$. Thus, any key in $S \cup \{x\}$ has the probability of $\frac{1}{n+1}$ for becoming the root of T'. The independence of left and right subtrees of T' follows from independence of left and right subtrees of T, induction, and the previous theorem.

13.7.2 Deletion in RBST

Suppose $x \in T$ and let T_x denote the subtree of T rooted at x. Assume that L and R are the left and right subtrees of T_x. To delete x, we build a new BST $T'_x =$ Join(L, R)

containing the keys in L and R and replace T_x by T'_x. Since pseudocode for deletion is easy to write, we omit the details.

We shall take a closer look at the details of the *Join* routine. We need couple of more notations to describe the procedure *Join*. Let L_l and L_r denote the left and right subtrees of L and R_l and R_r denote the left and right subtrees of R, respectively. Let a denote the root of L and b denote the root of R. We select either a or b to serve as the root of T'_x with appropriate probabilities. Specifically, the probability for a becoming the root of T'_x is $\frac{m}{m+n}$ and for b it is $\frac{n}{n+m}$ where $m = \text{size}(L)$ and $n = \text{size}(R)$.

If a is chosen as the root, the its left subtree L_l is not modified while its right subtree L_r is replaced with *Join*(L_r, R). If b is chosen as the root, then its right subtree R_r is left intact but its left subtree R_l is replaced with *Join*(L, R_l). The join of an empty tree with another tree T is T it self and this is the condition used to terminate the recursion.

```
Algorithm Join(L,R)
-- L and R are RBSTs with roots a and b and size m and n respectively.
-- All keys in L are strictly smaller than all keys in R.
-- $L_l$ and $L_r$ respectively denote the left and right subtree of L.
-- $R_l$ and $R_r$ are similarly defined for R.

1. If ( L is NULL) return R.
2. If ( R is NULL) return L.
3. Generate a random integer i in the range [0, n+m-1].
4. If ( i < m )   {* the probability for this event is m/(n+m).*}
      L_r = Join(L_l,R);
      return L;
   else           {* the probability for this event is n/(n+m).*}
      R_l = Join(L,R_l);
      return R;
```

It remains to show that *Join* of two RBSTs produces RBST and deletion preserves randomness in RBST.

THEOREM 13.17 *The Algorithm Join(L,R) produces a RBST under the conditions stated in the algorithm.*

Proof We show that any key has the probability of $1/(n+m)$ for becoming the root of the tree output by *Join(L,R)*. Let x be a key in L. Note that x will be at the root of *Join(L,R)* iff

- x was at the root of L before the *Join* operation, and,
- The root of L is selected to serve as the root of the output tree during the *Join* operation.

The probability for the first event is $1/m$ and the second event occurs with probability $m/(n+m)$. As these events are independent, it follows that the probability of x at the root of the output tree is $\frac{1}{m} \cdot \frac{m}{n+m} = \frac{1}{n+m}$. A similar reasoning holds good for the keys in R.

Finally,

THEOREM 13.18 *If T is a RBST for a set K of keys, then Delete(x,T) outputs a RBST for $K - \{x\}$.*

Proof We sketch only the chance counting argument. The independence of the subtrees follows from the properties of Join operation and induction. Let $T' = Delete(x, T)$. Assume that $x \in K$ and size of K is n. We have to prove that for any $y \in K, y \neq x$, the probability for y in root of T' is $1/(n-1)$. Now, y will be at the root of T' iff either x was at the root of T and its deletion from T brought y to the root of T' or y was at the root of T(so that deletion of x from T did not dislocate y). The former happens with a probability $\frac{1}{n} \cdot \frac{1}{n-1}$ and the probability of the later is $\frac{1}{n}$. As these events are independent, we add the probabilities to obtain the desired result.

13.8 Bibliographic Remarks

In this chapter we have discussed two randomized data structures for Dictionary ADT. Skip Lists are introduced by Pugh in 1990 [6]. A large number of implementations of this structure by a number of people available in the literature, including the one by the inventor himself. Sedgewick gives an account of the comparison of the performances of Skip Lists with the performances of other balanced tree structures [10]. See [7] for a discussion on the implementation of other typical operations such as merge. Pugh argues how Skip Lists are more efficient in practice than balanced trees with respect to Dictionary operations as well as several other operations. Pugh has further explored the possibilities of performing concurrent operations on Skip Lists in [8]. For a more elaborate and deeper analysis of Skip Lists, refer the PhD thesis of Papadakis [4]. In his thesis, he has introduced deterministic skip lists and compared and contrasted the same with a number of other implementations of Dictionaries. Sandeep Sen [5] provides a crisp analysis of the structural properties of Skip Lists. Our analysis presented in this chapter is somewhat simpler than Sen's analysis and our bounds are derived based on different tail estimates of the random variables involved.

The randomized binary search trees are introduced by Martinez and Roura in their classic paper [3] which contains many more details than discussed in this chapter. In fact, we would rate this paper as one of the best written papers in data structures. Seidel and Aragon have proposed a structure called probabilistic priority queues [11] and it has a comparable performance. However, the randomness in their structure is achieved through randomly generated real numbers(called priorities) while the randomness in Martinez and Roura's structure is inherent in the tree itself. Besides this being simpler and more elegant, it solves one of the outstanding open problems in the area of search trees.

References

[1] T.Hagerup and C.Rub, *A guided tour of Chernoff bounds*, Information processing Letters, 33:305-308, 1990.

[2] H.M.Mahmoud, *Evolution of Random Search Trees*, Wiley Interscience, USA, 1992.

[3] C.Martinez and S.Roura, *Randomized Binary search Trees*, Journal of the ACM, 45:288-323, 1998.

[4] Th.Papadakis, *Skip List and probabilistic analysis of algorithms*, PhD Thesis, University of Waterloo, Canada, 1993. (Available as Technical Report CS-93-28).

[5] S.Sen, *Some observations on Skip Lists*, Information Processing Letters, 39:173-176, 1991.

[6] W.Pugh, *Skip Lists: A probabilistic alternative to balanced trees*, Comm.ACM, 33:668-676, 1990.

[7] W.Pugh, *A Skip List cook book*, Technical report, CS-TR-2286.1, University of Maryland, USA, 1990.

[8] W.Pugh, *Concurrent maintenance of Skip Lists*, Technical report, CS-TR-2222, University of Maryland, USA, 1990.

[9] P.Raghavan and R.Motwani, *Randomized Algorithms*, Cambridge University Press, 1995.

[10] R.Sedgewick, *Algorithms*, Third edition, Addison-Wesley, USA, 1998.

[11] R.Seidel and C.Aragon, *Randomized Search Trees*, Algorithmica, 16:464-497,1996.

College of Engineering, Pune

(An Autonomous Institute of Govt. of Maharashtra, Permanently Affiliated to S.P. Pune University)

Department of Computer Engineering and I.T.

Curriculum Structure & Detailed Syllabus (UG Program)

Third Year B. Tech. Computer Engineering

(Effective from: A.Y. 2021-22)

B.Tech. Computer Engineering

Programme Educational Objectives (PEOs):

I. To create graduates with sufficient capabilities in computer engineering who can become researchers, entrepreneurs and software professionals to satisfy the needs of the core industry, research, academia and society at large.

II. To build ability to continuously learn the latest trends in computer engineering and engage in lifelong learning process.

III. To build engineers aware of professional ethics of the software Industry, and equipped with basic soft skills essential for working in community and professional teams.

Programme Outcomes (POs):

At the end of the program, the graduates will

a. Demonstrate knowledge in fundamentals of computer engineering

b. Have knowledge of the best practices in software engineering, project management and professional work environments.

c. Be aware of professional ethics, environmental and sustainability issues.

d. Able to demonstrate the ability to design creative solutions to real life and most relevant problems faced by the industry and society at large.

e. Able to communicate technical topics in written and verbal forms.

f. Demonstrate their ability to use the state of the art technologies and tools including Free and Open Source Software tools in developing software.

g. Demonstrate good performance in the competitive examinations for higher education.

h. Have the ability for lifelong self-learning.

Programme Specific Outcomes (PSOs)

Students will be able to

1. Demonstrate competence in Programming Technologies.

2. Design, implement, test software solutions in core Computer Engineering areas including Computer Networks, Databases, Systems Software, Computer Architecture, Artificial Intelligence, Software Engineering

3. Acquire and demonstrate skills in emerging area like Information Security, Data Science, Natural Language Processing, Cloud Computing, etc.

Correlation between the PEOs and the POs

PO→ PEO↓	a	b	c	d	e	f	g	h
I	✓	✓	✓		✓	✓	✓	✓
II	✓	✓	✓	✓	✓	✓	✓	✓
III		✓	✓	✓	✓			

Correlation between the PEOs and the PSOs

PSO→ PEO↓	1	2	3
I	✓	✓	
II	✓	✓	
III	✓	✓	

List of Abbreviations

Sr. No.	Abbreviation	Stands for:
1	BSC	Basic Science Course
2	DEC	Departmental Elective Course
3	HSMC	Humanities, Social Sciences and Management Course
4	IFC	Interdisciplinary Foundation Course
5	IOC	Interdisciplinary Open Course
6	LC	Laboratory Course
7	MLC	Mandatory Learning Course
8	PCC	Programme Core Course
9	SBC	Skill Based Course

Table of Contents

CURRICULUM STRUCTURE OF T. Y. B. TECH

Effective from A. Y. 2020-21

Semester V

Sr. No.	Course Type	Course Code	Course Name	Teaching Scheme			Credits
				L	T	P	
1	BSC	CT-21008	Probability and Statistics for Engineers	2	1	0	3
2	MLC	ML-21001	Constitution of India	1	0	0	0
3	HSMC	HS-21001	Entrepreneurship Principles and Process	1	0	0	1
4	HSMC	AS (HS)-2100X	Humanities Open Course - I • English Language Proficiency • Japanese Language • German Language • Etc	2	0	0	2
5	SBC	CT-21009	Software Engineering: Mini Project - Stage 1	0	1	2	2
6	IFC	PE(IF)-21001	Interdisciplinary Foundation Course – III Robotics(Manufacturing Department)	2	0	0	2
7	PCC	CT-21007	Computer Organization	3	0	0	3
8	PCC	CT-21001	Database Management Systems	3	0	0	3
9	LC	CT-21002	Database Management Systems Laboratory	0	0	2	1
10	PCC	CT-21003	Artificial Intelligence	3	0	0	3
11	LC	CT-21004	Artificial Intelligence Laboratory	0	0	2	1
12	PCC	CT-21005	Computer Networks	3	0	0	3
13	LC	CT-21006	Computer Networks Laboratory	0	0	2	1
			Total	20	2	8	25
				30			
			Total Academic Engagement and Credits				**Max. 25**

Minor / Honours courses: Mentioned separately

List of IFC Courses offered to other departments

SN	Course Title	Offered to Department	L	T	P	Credits
1	Data Analytics	Mechanical Engineering	1	0	2	2
2	Fundamentals of Machine Learning	Instrumentation Engineering	1	0	2	2
3	Fundamentals of Operating Systems	Instrumentation Engineering Electrical Engineering	2	0	0	2

Semester VI

Sr. No.	Course Type	Course Code	Course Name	Teaching Scheme			Credits
				L	T	P	
1	MLC	ML-21002	Environmental Studies	1	0	0	0
2	HSMC	AS (HS)-2100X	Humanities Open Course - II • Finance for Engineers • Engineering Economics • Industrial Psychology • Etc.	2	0	0	2
3	SBC	CT-21015	Mini project ["D-S-P-T: Design-Simulate- Prototype - Test "]: Software Engineering: Mini project - Stage II	2	0	2	3
4	IOC	IOC-21002	Interdisciplinary Open Course-1 Introduction to Artificial Intelligence	1	0	2	2
5	DEC		Department Elective-I	3	0	0	3
6	LC		Department Elective-I Laboratory	0	0	2	1
7	PCC	CT-21010	Operating Systems	3	0	0	3
8	LC	CT-21011	Operating Systems Laboratory	0	0	2	1
9	PCC	CT-21014	Design and Analysis of Algorithms	3	1	0	4
10	PCC	CT-21012	Data Science	3	0	0	3
11	LC	CT-21013	Data Science Laboratory	0	0	2	1
			Total	18	1	10	
			Total Academic Engagement and Credits	29			23
							Max. 25

Minor / Honours courses: Mentioned separately on next page

Department Elective-I : List of courses

Advanced Data Structures
Advanced Microprocessors
Web Systems and Technologies
Computer Graphics
Digital Signal and Image Processing
Parallel Computer Architecture and Programming
Computational Geometry
Recent Trends in Computer Networks
Courses in association with Industry

Minor in Computer Engineering

(To be offered to students of other departments)

SN	Semester	Course Name	Lectures-Tutorial-Lab-Credits
1	V	Data Structures, Files and Algorithms	3-0-0-3
2	VI	Object Oriented Programming and Design	3-0-0-3

Honours in Data Science (for students of Computer Engineering)

SN	Semester	Course Name	Lectures-Tutorial-Lab-Credits
1	V	Making Sense of Data	3-0-0-3
2	VI	Big Data Analytics	3-0-0-3

Honours in Information Security (for students of Computer Engineering)

SN	Semester	Course Name	Lectures-Tutorial-Lab-Credits
1	V	Fundamentals of Information and Coding Theory	3-0-0-3
2	VI	Ethical Hacking	3-0-0-3

(CT-21008) Probability and Statistics for Engineers

Teaching Scheme
Lectures: 2Hrs/ Week
Tutorials: 1 Hr / Week

Evaluation Scheme
Internal Test 1: 20 marks
Internal Test 2: 20 marks
End Semester Exam: 60 marks

Course Outcomes

Students will be able to:

1. Demonstrate number of methods of summarizing and visualizing data sets, evaluate probabilities of events.
2. Make use of concepts of random variables and associated probability distributions to solve problems, illustrate the central limit theorem.
3. Test for basic statistical inference (t-test, z-test, F-test, χ^2 –test, confidence interval, non parametric tests).
4. Explain basic principles of regression analysis and perform the same.
5. Demonstrate use of R software for all the above.

Course Contents

Descriptive statistics: Measures of location and variation. Visualization of data: Frequency tables, bar diagrams, histograms, heat maps, other visualization tools. Review on introduction to combinatorics and probability theory

[5 Hrs]

Some of the basic probability distributions: Binomial, Poisson, Exponential, and Normal. Central limit theorem.

[5 Hrs]

Introduction to 'R': Introductory R language fundamentals and basic syntax, major R data structures, Using R to perform data analysis, creating visualizations using R.

[4 Hrs]

Basic statistical inference and hypothesis testing: Estimation, basic tests such as t-test, z-test, F-test, χ2 –test; Non parametric tests: Sign test, Wilcoxon signed rank test.

[6 Hrs]

Regression methods: Simple linear regression and multiple regression.

[4 Hrs]

Engineering applications of statistics: Engineering applications of statistics (Branch Specific (any 2)): Discussion on reliability and quality control. Introduction to random processes, stochastic processes, Markov chains. Machine learning and data science.

[4 Hrs]

Text Books

- Ronald E, Walpole, Sharon L. Myers, Keying Ye, Probability and Statistics for Engineers and Scientists (8th Edition), Pearson Prentice Hall, 2007.
- Tilman M. Davies, The book of R: A first course in Programming and Statistics (1st Edition), No Starch Press, USA, 2016

Reference Books

- Ross S.M., Introduction to probability and statistics for Engineers and Scientists (8th Edition), Elsevier Academic press, 2014.
- S. P. Gupta, Statistical Methods, S. Chand & Sons, 37th revised edition, 2008.
- Kishor S. Trivedi, Probability and Statistics with Reliability, Queuing and Computer Science Applications (2nd Edition), Wiley Student edition, 2008.
- Stephens L.J., Schaum's outline of statistics for Engineers, Latest edition, 2019.
- The practice of Business Statistics by Manish Sharma and Amit Gupta, Khanna Publishing Company Private Limited, New Delhi, 2014.

References for R Software:

- Norman Matloff, The Art of R Programming - A Tour of Statistical Software Design, (1st Edition), No Starch Press, USA, 2011.
- Sudha Purohit, Sharad Gore, Shailaja Deshmukh, Statistics using R (2nd Edition), Narosa Publications, 2019.
- Randall Pruim, Foundations and Applications of Statistics - An introduction using R (2nd Edition), American Mathematical Society, 2018.
- Hadley Wickham and Garrett Grolemund, R for Data Science: Import, Tidy, transform, Visualize and Model Data, (1st Edition), O'Reilly Publications, 2017

(ML-21001) Constitution of India

Teaching scheme
Lectures: 1hr / week

Evaluation scheme
T1: 20 marks
T2: 20 marks
End Semester 60 Marks

Course Outcomes

At the end of the course, student will demonstrate the ability to:

1. Interpret the Preamble and know the basics of governance of our nation.
2. Identify the different aspects covered under the different important Articles.
3. Apprehend the basic law, its interpretation and the important amendments.
4. Understand our Union and State Executive better.
5. Recognize the basic that along with enjoying the rights one needs to fulfill one's duties.
6. Summarize and Gain confidence on our Constitution by knowing it better.

Course Contents

Unit I
Understanding the concept 'Rule of Law '
Meaning and history of Constitution.
Introduction to The Constitution of India, understanding its objects.
Preamble to the constitution of India

[5 Hrs]

Unit II
Understanding the concept of Human Rights and Fundamental Rights.
Fundamental rights under Part – III, exercise of the Rights, limitations and important cases.
Prerogative Writs.
Fundamental duties & their significance.

[4 Hrs]

Unit III
Relevance of Directive principles of State Policy.
Legislative, Executive & Judiciary (Union and State)
Constitutional Provisions for Scheduled Castes, Scheduled Tribes, & Backward classes.
Constitutional Provisions for Women & Children

[4 Hrs]

Unit IV
Emergency Provisions.
Electoral procedure in India
Amendment procedure and few important Constitutional Amendments

[2 Hrs]

Text Books

- Introduction to the Constitution of India by Durga Das Basu (Students Edn.) Prentice – Hall EEE, 19th/20th Edn..
- Engineering Ethics by Charles E.Haries, Michael. S.Pritchard and Michael J.
- Robins Thompson Asia

Reference Books

- An Introduction to Constitution of India by M.V. Pylee, Vikas Publishing

(HS-21001) Entrepreneurship Principles and Process

Teaching Scheme
Lectures: 1 hrs / week

Examination Scheme
Field Work/Assignments 40 marks
End Sem. Exam: 60 marks

Course Outcomes

At the end of the course, students will demonstrate the ability to:

1. Discover, develop, and assess different types of Entrepreneurial ventures and opportunities.
2. Learn about opportunity and risk analysis

3. Use the strategies for valuing your own company, and how venture capitalist and angel investors use valuations in negotiating milestones, influence and control
4. Pick correct marketing mix and how to position the company in the market by using analytical tools
5. Learn how to sale themselves and the product/service and to handle objections
6. Know how organizations operates, their process matrices, start new ventures, write winning business plans

Course Contents

Unit I

Market Research, Types of Companies and Organizations

Introduction to Entrepreneurship, Profile of the Entrepreneur, Market Gap /Opportunity Analysis, Market Research Methods, Defining the Focal Market: Market Segmentation, Industry analyzing– Research /Competitive Analysis. Company/ Organization Types, Legal Aspects, Taxation, Government Liaison, Building the Team, Mergers and Acquisitions

[3 Hrs]

Unit II

Business Finance, Marketing & Digital Marketing

Shares and Stakes, Valuation, Finance Creation (Investors/Financers), Revenue Plans and Projections, Financial Ratios, Business Lifecycle, Break Even. Marketing Basics, Marketing Strategy and Brand Positioning, Plans and Execution Techniques, Marketing Analytics, Online Marketing

[4 Hrs]

Unit III

Sales & Operations Management

Understanding Sales, Pitching Techniques, Sales strategies, Inside Sales v/s Outside Sales, RFP

Operational Basics, Process Analysis, Productivity, Quality

[3 Hrs]

Unit IV

Start-ups

Start-up Basics, Terms, Start-up Financing, Start-up Incubation, Start-up Incubation, Getting Listed

[2 Hrs]

Text Books

- The Startup Playbook: Secrets of the Fastest-Growing Startups From Their Founding Entrepreneurs by David Kidder
- True North by Bill George and Peter Sims
- Cardullo,M.W.P.E.(1999).Technological entrepreneurism: Enterprise formation, financing, and growyh. England: Research Studies Press Ltd.

Reference Books

- Kanungo,R.N.(1998). Entrepreneurship and innovation: Models for development (Ed.,Vol.2). New Delhi: Sage.
- Van Nostrand. Verma , J.C.,& Singh ,G.(2002).Small business and industry: A hand book for entrepreneurs. New Delhi: Response-Sage.
- Richard A Brealy & Steward C Myres. Principles of Corporate Finance, McGrawHills, 7th Edn, 2004

- Prasanna Chandra, Financial Management: Theory and Practice,TataMcGrawHills, 6th Edn, 2004I MPandey, Financial Management, Vikas Publishing

Humanities and Social Sciences Open Courses-I

(AS (HS)-21001) English Proficiency Language

Teaching scheme
Lectures: 2 hrs / week

Evaluation scheme
T1&T2: 60 Marks
End Semester: 40 Marks

Course Outcomes

At the end of the course, student will demonstrate the ability to:

1. understand concepts of English language and apply them practically.
2. reproduce meaningful and well-structured sentences for conversation or speech in English.
3. analyze, comprehend and write well and effectively produce enhanced formal communication in English.
4. display their Presentation skills and participate and produce healthy discussions both formally and informally among peers using English.
5. create impact by acquiring professional skills, confidently face interviews and be better employable and industry ready.

Unit 1
English for communication
Basic understanding of language and its need for effective business communication for Engineers, Formal and informal expressions, Vocabulary Building, Business Idioms

[8 Hrs]

Unit 2
Presentation Skill Development
Oral Presentations, Basic Mannerisms and Grooming required for professionals, Cross cultural communication, Business Etiquette

[6 Hrs]

Unit 3
Business Writing
Writing Mechanics, Note making, Summarizing, Letter &Email Writing, Business Reports, Statement of Purpose

[8 Hrs]

Unit 4
Employability Enhancement
Job Readiness, Interview Skills and Mock Interviews

[6 Hrs]

Reference books

- Business Communication by ShaliniVerma (2nd Edition) (Vikas Publishing House)
- Communication for Business: A Practical Approach by Shirley Tailor (Longman)

- Communication Skills for Engineers by S. Mishra & C. Muralikrishna (Pearson)
- Communication Skills for Technical Students by T.M. Farhathullah (Orient Longman)
- Enhancing Employability at Soft Skills by Shalini Varma (Pearson)
- Written Communication in English by Saran Freeman (Orient Longman)
- Corporate Communication by JaishriJethwaney (Oxford University Press)
- Business Correspondence and Report Writing, R. C. Sharma & Krishna Mohan (Tata McGraw Hill)
- Essential English Grammar (Intermediate&Advanced) Raymond Murphy (CUP)

(AS (HS)-21002) German Language

Teaching scheme
Lectures:2 hrs / Week

Evaluation scheme
Assignments: 40 Marks
End Sem. Exam: 60 Marks

Course Outcomes

At the end of the course, student will demonstrate the ability to:

1. acquire knowledge of facts about Germany and German culture (cultural sensitization).
2. adapt pronunciation of German letters and greetings.
3. identify and calculate numerical till 1000.
4. describe themselves and third person.
5. construct simple questions or sentences and interact with the teacher and classmates.
6. comprehend time and time related phrases, illustration of the same in conversations.
7. handle day to day situations like placing an order in the restaurant or interact with shopkeeper in the supermarket.

Unit 1
Guten Tag! (Good day)
Greetings, self introduction and partner introduction, numbers till 100, how to mention telephone number and email address, about countries, nationalities and languages.

[6 Hrs]

Unit 2
Freunde, Kollegen und ich (Friends, colleagues and myself)
Hobbys, days of the week, months, seasons and professions, classroom objects and classroom communication

[6 Hrs]

Unit 3
Dining out
Understanding German cusine, meal courses, names of the ingredients, conversation with the waiter and in the supermarket.

[6 Hrs]

Unit 4
Uhrzeit (Timing)
Mention time, daily routine, making appointments

[6 Hrs]

Unit 5
Grammatik (grammar)
Vocab, Verb conjugations, WH-question, verbs, pronounciation, personal pronouns, articles, Singular and Plural, negation.

[6 Hrs]

Reference Books

- Dengler.S., Rusch. P., Schmitz.S., & Sieber.T. Netzwerk, Deutsch als Fremdsprache. 2015. Goyal Publishers & Distributors Pvt. Ltd. Delhi, India
- You tube video series "learn German", "easy German" etc.
- Funk.H., Kuhn.C., & Demme.S. Studio d A1. Deutsch als Fremdsprache. 2011. Goyal Publishers & Distributors Pvt. Ltd. Delhi, India.

(AS (HS)-21003) Japanese Language

Teaching scheme
Lectures: 2 hrs / Week

Evaluation scheme
Assignments: 40 Marks
End Semester: 60 Marks

Course Outcomes

At the end of the course, student will demonstrate the ability to:

1. acquire knowledge of facts about Japan and Japanese culture,
2. familiarize with pronunciation of Japanese letters and daily greetings, Accent, Intonation and Japanese writing System Hiragana, Katakana and Kanji
3. identify numbers, Colors, Years,Months and Days, Time expressions, Directions to read the city map
4. describe themselves and third person and family members
5. construct simple questions or sentences and interact with the teacher and classmates.
6. apply Engineering Terminology and Japanese work culturesuch as Monozukuri, 5S, Kaizen, 3M, 5W1H etc.

Unit 1
Introduction to Japanese Language (Nihongo)
Recognize Japanese Characters Hiragana. Can read /write Hiragana script
Use basic classroom expressions
Exchange greetings Can thank someone or apologize someone
Recognize Japanese Characters Katakana Can read /write Katakana script
Can ask someone to say something again if you don't really understand

About Me & Food
Give simple self introduction Can ask and answer where you live and your age.
Can write your name, nationality, date of birth and occupation in Japanese.
Recognize the parts of a business card
Talk someone briefly about your family using a family photo and answer simple questions such as who is that? Number of family members.
Talk about your favorite foods you like and dislike. Talk about your breakfast.
Can respond when offered a drink. For example saying what you want to drink.
Can look at menu in a fast food restaurant and understand what is available.

Can look at different restaurants' signboards and understand what each place is.

[6 Hrs]

Unit 2

Home & Daily life

Say what kind of house you live in. Say what you have in your home.
Write an e mail inviting someone to your home. Visit/ Welcome a friend.
Ask /say where to put things in the room. Can read the buttons on an electric appliance
Can listen to a simple explanation when being shown around a room and understand the layout.
Recognize the name and address on signs. Talk about your daily routine. Say the time you do something.Talk about your schedule at work for the week.
Can listen to short and simple instructions at work and understand what to do.
Can read a simple, handwritten note at work and understand the instructions.
Can ask someone to lend you something at work .
Can look at a list of equipment and confirm if you have all the items.

[6 Hrs]

Unit 3

Holidays and Days off 1 and Towns

Can give a simple answer when asked about your hobbies and favorite things to do .
Talk about what you do on your days off.
Can read an event poster and find the important information such as the date, time and place.
Can ask and answer questions about whether you are going to an event etc.
Can say when you are available, when you are inviting someone to something or being invited
Recognize station and Taxi signs. How to get to particular destination using a map
Can say how you go to work and how long it takes.
Describe places in town and location
Can look at common signs in a station and understand what they mean.

[7 Hrs]

Unit 4

Shopping & **Holidays and Days off 2**

Talk about what you want to buy.
Can ask staff in a shopping center etc .Where to go for a certain item and understand the answer .
Can look at discount signs and read the prices.
Make a brief comment on things in a shop.
Can read a short blog / simple e mail
Can talk in simple terms about impressions of the holiday / trip .
Can write a simple post for social media etc . About what you did in holiday.

[6 Hrs]

References Books

- Marugoto A1 Katsudo Starter Coursebook for Communicative Language Activities.
- Marugoto A1 Rikai Starter Coursebook for Communicative Language Competences
- The Japan Foundation
- Minna no Nihongo Main Textbook Elementary Lesson 1-12
- Minna no Nihongo Translation & grammatical Notes in English Elementary Lesson 1-12,3A Corporation Goyal Publishers

(AS (HS)-21004) Spanish Language

Teaching scheme
Lectures: 2 hrs / Week

Evaluation scheme
Assignments 40 Marks
End Semester 60 Marks

Course Outcomes

At the end of the course, student will demonstrate the ability to:

1. acquire knowledge of facts about Spain and Latin Americaand Spanish culture, pronunciation of Spanish letters and greetings.
2. identify and calculate numerical till 1000.
3. describe themselves and third person.
4. construct simple questions or sentences and interact with the teacher and classmates.
5. comprehend time and time related phrases, illustration of the same in conversations,
6. handle day to day situations like placing an order in the restaurant or interact with shopkeeper in the supermarket.

Unit 1
Hola! (Hello)
Greetings, self introduction and partner introduction, numbers till 100, how to mention telephone number and email address, about countries, nationalities and languages. Hobbys, days of the week, months, seasons and professions, classroom objects and classroom communication.

[6 Hrs]

Unit 2
La comida (Food)
Understanding Spanish cusine, meal courses, names of the ingredients, converstaion with the waiter and in the supermarket.

[6Hrs]

Unit 3
La ropa (clothing)
Clothing, accessory (as per weather), season + weather, vocabulary, Demonstrative pronouns, how to ask about price, numbers till 1000 .

[6 Hrs]

Unit 4
La hora (Timing)
Mention time, daily routine, making appointments

[6 Hrs]

Unit 5
La gramática (grammar)
Vocab, Verb conjugations, WH-question, verbs, pronounciation, personal pronouns, articles, Singular und Plural, negation.

[6 Hrs]

Reference Books

- Aula internacional 1Jaime Corpas, Eva García, AgustínGarmendia, Neus Sans Baulenas (contributor), published by Goyal Publisher's and Distributors Pvt. Ltd.

(CT-21009) Software Engineering (Mini Project) Stage- I

Teaching Scheme:
Laboratory: 2 Hrs/Week
Tutorial: 1 Hr / Week

Examination Scheme:
Continuous evaluation: 70 Marks
End Semester Exam: 30 Marks

Course Outcomes

Students will be able to-

1. Demonstrate the use of tools and technologies used in software project development process.
2. To expose students to FOSS environment and introduce them to use open source tools in developing software.
3. Demonstrate the ability to communicate, solve technical problems, work in teams, and contribute to an ongoing software project.
4. Write test cases for a specified task

Text Books/ Study Material/ Web Resources:

- "Debian New Maintainers' Guide", www.debian.org/doc/manuals/maint-guide
- Pro Git Book https://git-scm.com/book
- Autotools, GNU Manuals www.gnu.org/software/autoconf/
- GNU Gettext Manual https://www.gnu.org/software/gettext/manual/gettext.html
- Advanced Bash Scripting Guide, http://tldp.org/LDP/abs/html/

Suggested List of Assignments

- Write shell scripts for following tasks: convert a CSV file to VCF format, convert a youtube transcript to SRT format, find the top 10 size files created in last 20 days, move all duplicate files (except one) from a folder to a target location, etc.
- Write shell scripts or scripts in any language of your choice, to run conformance tests on a software of your choice.
- Create a git remote repository on any of the git hosting websites, using one of the software you have written so far. In a group of three or more people, carry out the following activities: reporting of bus, assigning of issues, fixing bugs, git branch and git pull requests.
- Localise and/or Internationalize any software and demonstrate your contributions. You may select any existing free software project for the same.
- Configure any of your existing C projects of atleast 500 lines using Autotools or Cmake or scons or any similar tool. You should write the required configuration files (like configure.in, Makefile.am files etc.) and also write a bootstrap program if needed.
- Package your software for Debian, Ubuntu, any Unix or other operating systems. For free software operating systems you should get your packaged software accepted by the respective communities.

- Fix bugs in any existing software, preferably a open source software by participating in the community development process.

This list is a guideline. The instructor is expected to improve it continuously.

IFC courses

(CT (IF)-21001) Data Analytics

Teaching Scheme	**Examination Scheme**
Lectures: 1 Hr/week	Continuous Lab/Project assessment: 40 marks
Laboratory: 2 Hr/wee	Midsem Exam: 30 Marks End-Sem Exam: 30 Marks

Course Outcomes

Student will be able to:

1. Examine and compare various datasets and features.
2. Analyze the business issues that analytics can address and resolve.
3. Apply the basic concepts and algorithms of data analytics.
4. Interpret, implement, analyse, and validate data using popular data analytics tools.

Course Contents

Fundamentals of Data Analytics: Descriptive, Predictive, and Prescriptive Analytics, Data Types, Analytics Types, Data Analytics Steps: Data Pre-Processing, Data Cleaning, Data Transformation, and Data Visualization. **[2 Hrs]**

Data Analytics Tools: Data Analytics using Python: Statistical Procedures, NumPy, Pandas, SciPy, Matplotlib

[2 Hrs]

Data Pre-Processing: Understanding the Data, Dealing with Missing Values, Data Formatting, Data Normalization, Data Binning, Importing and Exporting Data in Python, Turning categorical variables into quantitative variables in Python, Accessing Databases with Python.

[2 Hrs]

Data Visualization: Graphic representation of data, Characteristics and charts for effective graphical displays, Chart types- Single var: Dot plot, Jitter plot, Error bar plot , Box-and-whisker plot, Histogram, Two variable: Bar chart, Scatter plot, Line plot, Log-log plot, More than two variables: Stacked plots, Parallel coordinate plot.

[2 Hrs]

Descriptive and Inferential Statistics: Probability distributions, Hypothesis testing, ANOVA, Regression

[2 Hrs]

Machine Learning Concepts: Classification and Clustering, Bayes' classifier, Decision Tree, Apriori algorithm, K-Means Algorithm, Logistics regression, Support Vector Machines, Introduction to recommendation system.

[4 Hrs]

Text Books

- Anil Maheshwari, "Data Analytics made accessible," Amazon Digital Publication, 2014.
- James R. Evans, "Business Analytics: Methods, Models, and Decisions", Pearson 2012
- Song, Peter X. K, "Correlated Data Analysis: Modeling, Analytics, and Applications", Springer-Verlag New York 2007.

Reference Books

- Glenn J. Myatt, Wayne P. Johnson, "Making Sense of Data I: A Practical Guide to Exploratory Data Analysis and Data Mining", Wiley 2009.
- Thomas H. Davenport, Jeanne G. Harris and Robert Morison, "Analytics at Work: Smarter Decisions, Better Results", Harvard Business Press, 2010
- Rachel Schutt, Cathy O'Neil, "Doing Data Science", O'REILLY, 2006.
- Shamanth Kumar Fred Morstatter Huan Liu "Twitter Data Analytics", Springer-Verlag, 2014.

List of Assignments

1. Write a NumPy program to generate an array of 15 random numbers from a standard normal distribution.
2. Write a NumPy program to create a two-dimensional array with shape (8,5) of random numbers. Select random numbers from a normal distribution (200,7).
3. Write a Pandas program to add, subtract, multiple and divide two Pandas Series. Sample Series: [2, 4, 6, 8, 10], [1, 3, 5, 7, 9]
4. Write a Pandas program to convert a NumPy array to a Pandas series.
5. Write a Pandas program to create the mean and standard deviation of the data of a given Series.
6. Write a Pandas program to compute the minimum, 25th percentile, median, 75th, and maximum of a given series.
7. Write a Pandas program to get the day of month, day of year, week number and day of week from a given series of date strings.
8. Consider Iris Dataset, load the iris data into a dataframe and perform following basic operations on it:
    a. print the shape of the data, type of the data and first 10 rows and get the number of observations, missing values and nan values.
    b. Use Scikit-learn to print the keys, number of rows-columns, feature names and the description of the Iris data.
    c. create a 2-D array with ones on the diagonal and zeros elsewhere. Now convert the NumPy array to a SciPy sparse matrix in CSR format
    d. basic statistical details like percentile, mean, std etc. of iris data.
    e. Write a Python program to drop Id column from a given Dataframe and print the modified part. Call iris.csv to create the Dataframe.
    f. create a plot to get a general Statistics of Iris data
9. Consider the same Iris Dataset and perform visualization on the same:
    a. Write a Python program to create a Bar plot and pie plot to get the frequency of the three species of the Iris data.
    b. Write a Python program to create a graph to see how the length and width of SepalLength, SepalWidth, PetalLength, PetalWidth are distributed.
    c. Write a Python program to create a joinplot to describe individual distributions on the same plot between Sepal length and Sepal width. Note: joinplot - Draw a plot of two variables with bivariate and univariate graphs.

d. Write a Python program to draw a scatterplot, then add a joint density estimate to describe individual distributions on the same plot between Sepal length and Sepal width.
e. Write a Python program using seaborn to Create a kde (Kernel Density Estimate) plot of sepal_length versus sepal width for setosa species of flower.
f. Write a Python program to create a box plot (or box-and-whisker plot) which shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable of iris dataset. Use seaborn.
g. Write a Python program to create a Principal component analysis (PCA) of iris dataset.

10. Write a Python program using Scikit-learn to split the iris dataset into 80% train data and 20% test data. Train or fit the data into the model and using the K Nearest Neighbor Algorithm and create a plot of k values vs accuracy.
11. Build a decision tree model that predicts the species of iris from the petal and sepal width and length. Perform model evaluation.
12. Implementing Support Vector Machine (SVM) classifier in Python using the iris features from iris dataset and train an SVM classifier and use the trained SVM model to predict the Iris species type.

Mini Project: Write an application demonstrating your skills in defining a data science problem, writing down the requirements carefully, designing a modular solution with clear separation of data pre-processing and transformation, visualization ,model building and model evaluation. The application can use any dataset from Kaggle, UCI etc or a task defined after discussion with the instructor.

This list is a guideline. The instructor is expected to improve it continuously.

(CT(IF)-21002) Fundamentals of Machine Learning

Teaching Scheme:
Lectures: 1 Hrs/week
Lab: 2Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will able to:

1. To introduce students to the basic concepts, tools and techniques of Machine Learning.
2. To develop skills of using recent machine learning software for solving practical problems.
3. Analyze and Evaluate the different ML models.
4. Implement ML algorithms to solve real life problems.

Course Contents

Prerequisites: Relevant applied math and statistics: probability theory, probability distribution, Conditional probability, Bayesian probabilities.

Introduction to Machine Learning: Basic concepts, Machine Learning methods: Supervised, Unsupervised, Semi-supervised, Inductive, Reinforcement Learning.

[2 hrs]

Linear Regression: Introduction toLinear regression, Logistic Regression, Naive BayesAlgorithm, Model Selection, Linear basis function model, model assessment, assessing importance of different variables, subset selection. Cross Validations.

[3 hrs]

Hypothesis Design: Types of variables, Types of measurement scales, Constructing the Hypothesis, Null hypothesis, Alternative Hypothesis. Hypothesis testing, type 1 error, Type 2 error, Confidence of Interval.

[3 hrs]

Instance Based Learning: Feature selection, supervised and unsupervised learning, Classification Algorithms: K-Nearest Neighbour Classification and Decision Tree.

[3 hrs]

Neural Network: Introduction, Feed forward network, Network training, Back propagation NN, Regularization, Error Analysis, Deep Neural Network.

[3 hrs]

Text Books

- Tom M. Mitchell, "Machine Learning", First Edition, McGraw Hill Education, ISBN 978-12-5909-695-2
- Andreas C. Müller and Sarah Guido , "Introduction to Machine Learning with Python: A Guide for Data Scientists", First Edition, O'Reilly Media, ISBN 978-14-4936-941-5

Reference Books

- Trevor Hastie, Robert Tibshirani, and Jerome Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Second Edition, Springer, ISBN 978-03-8784-857-0
- Christopher M. Bishop, "Pattern Recognition and Machine Learning", Second Edition, Springer, 978-03-8731-073-2
- Hadley Wickham and Garrett Grolemund, "R for Data Science: Import, Tidy, Transform, Visualize, and Model Data", First Edition, O'Reilly, ISBN 978-14-9191-039-9

List of Practical Assignments:

1. Exploratory Data Analysis: Perform following operations on any open dataset available in Python/Kaggle:
    a. Load data into a data frame from a .cvs or any other file format.
    b. Identification of variables and data types.
    c. Display number of rows and columns.
    d. Find Missing Values.
    e. Replace/eliminate missing values
    f. Change column name(s) to short/easy names if required.
    g. Drop unessential columns.
    h. Add new columns
    i. Display first/last few rows.

j. Find average/min/max of numeric columns.
k. Find mode of non-numeric columns.
l. Display unique values in each column.
m. Display summary of dataframe

2. Plots: Using various plots explore the relationship between attributes of a dataset.

3. Build a Linear Regression Model using *New York Stock Exchange* dataset, to predict. This dataset contains historical data from the New York stock market.

4. Build a Linear Regression Model using *Real estate price prediction* dataset.
This real estate dataset was built for regression analysis, linear regression, multiple regression, and prediction models. It includes the date of purchase, house age, location, distance to nearest MRT station, and house price of unit area.
Take a dataset contains 1000 or more row, and each input point contains 3 features. Train a linear regression model on the dataset. Report the coefficients of the best fit model.

5. Developed a logistic regression model to predict whether a produced microchip should be accepted or rejected based on a dataset from quality tests.

6. Create a Data frame having below contents:

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Use Decision Tree algorithm to predict to play or not to play.

7. Use a dataset like 'Credit.csv' contains data existing customers with below attributes

Variables	Variable Description
Cdur	Credit duration (in months)
Cpur	Purpose of credit (Business, electronics, etc.)
Camt	Credit amount (in □)
Prop	Properties held by the borrower (real estate, other cars, etc.)
age	Age of the borrower
creditScore	Credit rating of the borrower (bad/good)

Build a k-NN model for classifying the credit rating of the customers based on their credit history. What is the highest accuracy (in %) achieved by this k-NN model.

8. Use the 'iris.csv' dataset, which has four features of a flower, as mentioned below. Determining the optimum number of clusters using K-Means clustering algorithm. Build the algorithm and find the optimum 'k' value, ranging between 1 to 10 by setting the seed value as 111.

Variables	Variable Description
Sepal.Length	Sepal Parameters (Length and Width, in cm)
Sepal.Width	
Petal.Length	Petal Parameters (Length and Width, in cm)
Petal.Width	

What is the optimal number of clusters for the k-means model built using the information given above? Determine the value of between-clusters sum of squares of and the value of the total sum of squares for the k-means model built with the optimal k value.

9. **Train** a neural network classifier using the dataset (https://www.kaggle.com/datasets). Run cross-validation and compute the classification error per fold.
   Also perform a statistical test on the two sets of error observations (one from decision trees and one from neural networks) and report your findings.

10. Add some new emotions to the existing dataset. What changes should be made in decision trees and neural networks classifiers in order to include new classes? Justify

This list is a guideline. The instructor is free to assign new assignments.

(CT(IF)-21003) Fundamentals of Operating Systems

Teaching Scheme:
Lectures: 2 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Write programs to manipulate processes, files, and hardware resources using appropriate system calls.
2. Illustrate the design issues, solutions and complexity of operating system by compiling, modifying an OS kernel, tracing the sequence of activities on processor,

data structures of a file system, race conditions, locking mechanisms and storage techniques.
3. Correlate the computer architecture features with operating system design issues.
4. Make design choices for an operating systems with given constraints

Course Contents

Introduction and Operating Systems structures: System programs: compiler, linker, loader. Operating system components, O.S. Services, System Calls, Virtual Machines, Boot Sequence.

[3 Hrs]

Processes and CPU Scheduling: Process concept, interleaved I/O and CPU burst; Process states; Co-operating processes, Thread, Thread libraries, Multithreaded programming, Scheduling, Scheduling criterion, Scheduling algorithms, Interrupts and Interrupt handling.

[4 Hrs]

Inter process Communication: Pipes, Shared memory mechanism, , Asynchronous communication, Signals. POSIX API for IPC and programs using it.

[3 Hrs]

Process Synchronization: Critical section problem, Hardware support for mutual exclusion, Semaphores, Deadlock-principle, Deadlock detection, prevention and avoidance, Classical problems in concurrent programming: Producer-consumer, Reader-writer with and without bounded buffer. Design of locking primitives like spinlock, semaphore, read-write locks, recursive locks, etc.

[6 Hrs]

Memory management: O.S. and hardware interaction, Swapping, Continuous memory management, paging, Segmentation, Virtual Memory Management, Demand Paging, Page replacement algorithms, Allocation of frames, Kernel memory management

[6 Hrs]

File Management and Storage Structures: File Organization, Concept of files and directories, System calls for file systems, Free space management, Data structures like inode and super block, Virtual file system, Disk layout, Formatting, Recovery

[6 Hrs]

Text Books

- Abranhan Silberschatz, Peter B Galvin, Greg Gagne; Operating System Concepts, Wiley India Students Edition, 8th Edition, ISBN: 978-81-265-2051-0
- Andrew S. Tanenbaum; Modern Operating Systems; Prentice Hall of India Publication; 3rd Edition. ISBN: 978-81-203-3904-0

Reference Books

- Milan Milenkovic; Operating Systems; Tata McGraw Hill; Second Edition. ISBN: 0-07-044700-4

- Maurice J. Bach; The Design of the Unix Opearating System; Prentice Hall of India; ISBN: 978-81-203-0516-8
- Uresh Vahalia; Unix Internals, The New Frontiers; Prentice Hall; ISBN: 0-13-101908-2

List of Assignments

1. Create two virtual machines using virtual box software. One virtual machine will run a GNU/Linux of your choice on it. The other virtual machine will run any non-Linux operating system.
2. Write a minimal version of a shell with the features to handle pipes and handle redirection.
3. Use debugfs tool to locate a file which was recently deleted on an ext2 file system. Write a program, on the lines of debugfs, to browse an ext2 file system and given the complete name name of a file, print it's inode.
4. Write a program using pthreads to demonstrate the producer consumer synchronization problem. Implement appropriate synchronization. Show the different results with and without synchronization.
5. Write a program to demonstrate the usage of signals – show how processes can wait for each other, kill each other, stop and continue each other.

This list is a guideline. The instructor is free to assign new assignments.

(PE(IF)-21001) Robotics (Manufacturing Department)

Teaching Scheme:
Lectures: 2 Hrs/week

Examination Scheme:
Quizzes – 40 marks
End Sem Exam - 60 marks

Course Objectives:

1. To enable students to understand the basic concepts and principles in robotics.
2. To enable students to classify the robot structures, grippers, drives and their design and selection
3. To enable students about kinematics of robot manipulator and transformation analysis.
4. To enable students to understand robot programming and write the programs.
5. To enable students to analyze the trajectory planning of robot joints.
6. To select robots for various applications and perform economic analysis

Course Contents

Unit 1: Basic Concepts in Robotics: Automation and robotics, robot anatomy, basic structure of robots. Classification and Structure of Robotics System: Point to point and continuous path systems. Control loops of robotic system, manipulators, wrist motions and grippers.
Robot End Effectors: Grippers and tools, Types of end effectors-mechanical, magnetic and vacuum, gripper force analysis and gripper design considerations.

[5 Hrs]

Unit 2: Drives and Sensors: Basic control systems, concepts and models, types of drive system- Hydraulic systems, pneumatic and electrical, DC servo motors, analysis, robot activation and feedback components,

Sensors, internal-external sensors, contact and non-contact sensors, position and velocity sensors, Touch and slip sensors, Force and torque sensors, tactile sensors, Proximity and range sensors.

[6 Hrs]

Unit 3: Robot Arm Kinematics: Homogenous coordinates and homogenous transformations, Forward and Inverse kinematics in robot, Denavit Hartenberg convention and its applications Lagrange-Euler formation.

[5 Hrs]

Unit 4 : Robot Programming: Methods of robot programming, lead through, motion interpolation, WAIT, SIGNAL and DELAY commands, branching capabilities and limitations of lead through methods. Robot Language: The textual robot languages, generations of robot programming languages, variables, motion commands, end effectors and sensor commands, computations and operations.

[4 Hrs]

Unit 5: Trajectory Planning: Introduction, Joint Space Scheme, Cubic Polynomials with via points, Blending scheme

[4 Hrs]

Unit 6: Robot Applications in Manufacturing: Material transfer and machine loading/unloading, processing operations assembly and inspection. Concepts of safety in robotics, social factors in use of robots, economics of robots, Telechiric machines and its application.

[4 Hrs]

Text Books

- S. R. Deb.: Robotics Technology And Flexible Automation, Tata McGraw Hill Publishing Co. Ltd.
- P.A. Janakiraman, Robotics and Image Processing, Tata Mcgraw Hill, 1995

Reference Books

- Yoren Koren: Robotics for Engineers, McGraw Hill Book Co., ISBN 0-07-035341-7.
- M. P. Grover, M. Weiss, R. N. Nagel, N. G. Odrey,: Industrial Robotics Technology, ISBN 0-07-100442-4.
- K. S. Fu, C. G. S. Lee, R. C. Gonzaler, Robotics Control, Sensing, Vision and Intelligence, Tata McGraw Hill. 2008, ISBN 13: 9780070226258

(CT-21007) Computer Organization

Teaching Scheme
Lectures: 3Hrs/ Week

Examination Scheme
Assignment/Quizzes: 40 marks
End Semester Exam: 60 marks

Course Outcomes

Students will be able to:

1. Analyze Instruction set architecture, control signals in CPU, Hard wired control & Microprogrammed control units.

2. Apply Booth algorithm for multiplication, floating point number representation & Arithmetic.
3. Describe DRAM technology, cache memory,
4. Justify the need of paging in virtual memory and Secondary Storage.
5. Measure and analyze multiprocessor system & bus arbitration, Instruction pipelining & RISC.

Course Contents

CPU Architecture: Instruction format, control signals in CPU, micro program control unit and hardwired control unit, ALU & sequencer, look ahead carry generator, MIPS ISA

[6 Hrs]

Arithmetic: Integer Arithmetic-multiplication, Booth's Algorithm, division algorithm; Floating point number representation, and floating point arithmetic

[6Hrs]

Memory: Dynamic RAM organization, CACHE memory & it's mapping, cache coherence & MESI protocol, virtual memory, secondary storage, MBR and GPT hard disks, RAID, File system FAT

[8Hrs]

System and memory map: Closely coupled and loosely coupled multiprocessor systems, bus arbitration, co-processor, lower 1MB memory map

[7 Hrs]

Instruction Pipelining: Basic concepts and issues, Introduction to the basic features & architecture of RISC & CISC processors, super scalar processor, MIPS pipeline

[7Hrs]

Multiprocessor: Introduction to Multicores, Multiprocessors and Clusters. Introduction to GPGPU

[6Hrs]

Text Books

- William Stallings, Computer Organization and Architecture, 9/E ISBN-10: 013293633X ISBN-13: 9780132936330©2013 Prentice Hall
- Carl Hamacher, ZvonkoVraesic and SafwatZaky, Computer Organisation, ISBN 0-07-232086-9, MGH 5th edition.

Reference Books

- D. Paterson, J. Hennesy, "Computer Organization and Design: The Hardware Software Interface", 5th Edition MIPS, Morgan Kauffman, 2016 ISBN 9351073378.
- Liu & Gibson, Microcomputer Systems, PHI, ISBN: 978-81-203-0409-3

(CT-21001) Database Management Systems

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Identify and describe various components of DBMS.
2. Construct Entity-Relationship Model for given applications and Relational Model for the same.
3. Design and write best possible or optimal (SQL) query statement for given statement.
4. Apply normalization to database design.
5. Improve efficiency of data retrieval using various storage systems and indexing.
6. Describe concurrency control protocol and solve analytical problems on serializability.

Course Contents

Introduction: Basic concepts, Database system application, Purpose of database systems, View of data, Database languages; Database architecture: Components of DBMS and overall structure of DBMS; Types of databases

[6 Hrs]

Relational Model: Structure of relational databases, Fundamental relational algebra operations, Additional relational algebra operations, Extended relational algebra operations, null values, Modification to database, Other relational languages .

[6 Hrs]

SQL: Basic structure and operations, Aggregate functions, Nested subqueries, Complex queries, views; Cursors in SQL; No SQL databases: Features of NoSQL databases, Types of NoSQL databases.

[6 Hrs]

E-R model: Entity, Attributes, Relationships, Constraints, E-R model design; Relational Database Design: Basic concept of normalization, Decomposition using functional dependencies.

[8 Hrs]

Indexing and Hashing: Basic of query processing; Indices: Concepts, B+ trees and B-tree index file; Static and dynamic hashing.

[6 Hrs]

Transactions: Basic concepts, States, Concurrent execution, Serializability, Recoverability, isolation; Concurrency control: Timestamps and locking protocols, Validation based protocols, deadlock handling; Recovery**:** Log-based recovery, Shadow-paging.

[8 Hrs]

Text Books

- Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Database system concepts", Fifth Edition, McGraw Hill International Edition, ISBN 978-0073523323.
- Raghu Ramkrishnan, Johannes Gehrke, "Database Management Systems", Second Edition, McGraw Hill International Editions, ISBN 978-0072465631.

Reference Books

- Rob Coronel, "Database systems: Design implementation and management", Forth Edition, Thomson Learning Press, ISBN 978-1418835934.
- Ramez Elmasri and Shamkant B. Navathe, "Fundamental Database Systems", Third Edition, Pearson Education, 2003, ISBN 978-0321204486.
- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, "Database Systems: The Complete Book", Second Edition, Pearson, 2008, ISBN 978-0131873254.

(CT- -21002) Database Management Systems Laboratory

Teaching Scheme:
Laboratory: 2 Hrs/Week

Examination Scheme:
Continuous evaluation: 50 Marks
Oral Exam: 50 Marks

Course Outcomes

1. Explain fundamentals and various components of DBMS.
2. Analyze requirements and Design E-R data model by applying different forms of constraints on data for the given application/problem in hand.
3. Think and write best possible or optimal (SQL) query statement for given statement.
4. Explain normalization by applying Normalizations techniques to database .
5. Write high level program to connect database through appropriate API.

Suggested List of Assignments

1. Write simple SQL Queries on the given schema.
2. Write SQL queries using aggregates, grouping and ordering statements for given statements on given schema.
3. Write SQL queries for given schema using Nested Subqueries and SQL Updates
4. Write DDL and DML statements for given statements.
5. Create the schema and constraints on the given relations using given statements.
6. Demonstrate database connectivity through High Level Programming Language.
7. Demonstrate the application of cursor in database.
8. Select any real time problem for database implementation. Draw an ER diagram for the selected problem in hand. Normalise the database up to appropriate normal form.

This list is a guideline. The instructor is expected to improve it continuously.

(CT- -21003) Artificial Intelligence

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Compare AI with human intelligence and traditional information processing and discuss its strength and limitations
2. Apply the basic principles, models and algorithms of AI to recognize, model and solve problems
3. Demonstrate knowledge of basics of the theory and practice of Artificial Intelligence
4. Design and carry out an empirical evaluation of different algorithms on a problem formalisation, and state the conclusions that the evaluation supports.
5. Apply knowledge representation techniques and problem solving strategies to common AI applications.

Course Contents

Introduction: What is AI, History, AI problems, Production Systems, Problem characteristics, Intelligent Agents, Agent Architecture, AI Application (E-Commerce, & Medicine) , AI Representation, Properties of internal representation, Future scope of AI, Issues in design of search algorithms.

[6 Hrs]

Heuristic search techniques: Heuristic search, Hill Climbing, Best first search, mean and end analysis, Constraint Satisfaction, A* and AO* Algorithm**, Knowledge Representation:** Basic concepts, Knowledge representation Paradigms, Structured representation of knowledge, ISA hierarchy, Frame notation, Resolution, Natural Deduction

[6 Hrs]

Knowledge Inference: Introduction, Knowledge representation- Production based system, Forward and Backward reasoning, Knowledge representation using non monotonic logic: TMS (Truth maintenance system) , statistical and probabilistic reasoning, fuzzy logic, structure knowledge representation, semantic net, Frames, Script, Conceptual dependency.

[6 Hrs]

Learning & Planning: What is Learning, Types of Learning (Rote, Direct instruction Analogy, Induction, Deduction), Planning: Block world, strips, Implementation using goal stack, Non linear planning with goal stacks, Hierarchical planning, Least commitment strategy.

[6 Hrs]

Advanced AI Topics: Game playing: Min-max search procedure, Alpha beta cutoffs, waiting for Quiescence, Secondary search, Natural Language Processing (NLP): Introduction, Steps in NLP, Syntactic Processing, Semantic analysis, Discourse & Pragmatic Processing. Perception and Action: Perception, Action, Robot Architecture.

[8 Hrs]

Neural Networks and Expert systems: Neurons and biological motivation. Linear threshold units. Perceptrons: representational limitation and gradient descent training. Multilayer networks and backpropagation, Hidden layers and constructing intermediate, distributed representations, Overfitting, learning network structure, two case studies on expert systems.

[8 Hrs]

Text Books

- Elaine Rich and Kerin Knight, Artificial Intelligence, 3rd Edition, McGraw Hill. ISBN13: 9780070087705
- Eugene, Charniak, Drew Mcdermott, Introduction to artificial intelligence, Addison-Wesley. ISBN 0-07-052263-4.

Reference Books

- Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 3rd Edition. ISBN 0-13-103805-2.
- Herbert A. Simon, The Sciences of the Artificial, MIT Press, 3rd Edition, 1998. *ISBN*: 9780262190510.
  George F Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Pearson Edu., 4th Edition. *ISBN*-13: 978-0-321-54589-3.

(CT- 21004) Artificial Intelligence Laboratory

Teaching Scheme:
Laboratory: 2 Hrs/week

Examination Scheme:
Continuous evaluation: 50 Marks
End Semester Exam: 50 Marks

Course Outcomes

Upon successful completion of the course, the students will be able to:

1. Develop an Explaining what is involved in learning models from data.
2. Implement a wide variety of learning algorithms.
3. Apply principles and algorithms to evaluate models generated from data.
4. Apply the algorithms to a real-world problem.

Suggested List of Assignments

1. Implement A* algorithm .
2. Implement AO* algorithm .
3. Implementation of other Searching algorithm.
4. Implementation of Min/MAX search procedure for game Playing .
5. Implementation of variants of Min/ Max search procedure.
6. Implementation of mini Project using the concepts studied in the AI course.

This list is a guideline. The instructor is expected to improve it continuously.

(CT-21005) Computer Networks

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Trace the flow of a network packet over internet
2. Design a network with subnets as specified
3. Discuss and critically evaluate various networking protocols used in Internet
4. Discuss and analyze the problems posed by each layer of TCP/IP stack
5. Analytically compare the state-of-the-art protocols and architectures in computer networks for Internet

Course contents

Introduction to computer networks and Internet: Physical Mediums, Concept of Network Protocols, Layered Architecture, Network core – Packet Switching vs Circuit Switching, Various Losses in the Internet, ISPs, IXPs, Routing, Heterogeneous Networks, Brief History of Computer Networks

[4 Hrs]

Application Layer: Basics of Socket Programming, Transport Layer Programming Interface(TCP, UDP) , Protocols: HTTP (Overview, Persistent and Non-Persistent, Message Format, Cookies, Cachess) , SMTP (Overview, Message Formats) , IMAP, POP, DNS; FTP; Telnet, SSH; Peer-to-Peer Applications, BitTorrent Protocol; Conte Distribution Networks;

[8 Hrs]

Transport Layer: Relationship Between Transport and Network Layer, TCP and UDP; Multiplexing and Demultiplexing; Implementation Overview in OS kernels; Principles of Reliable Data Transfer; Go-Back-N and Selective Repeat; TCP: Segment Structure, Round Trip Time Estimation, Reliable Data Transfer, State Transitions, Flow Control, Congestion Control, Fairness; UDP: Segment Structure

[8 Hrs]

Network Layer, Subnets: Concept of IP Address, Netmask, Subnet; CIDR; Design of a LAN and WAN; Subnetting Problems

[3 Hrs]

Network Layer, Routers,IPv4, IPv6: Functions of a Router: Forwarding and Routing; Inside of a Router: Port Processing, Switching; Queueing: Causes, Delays; IPV4: Datagram Format, Fragmentation; Network Address Translation; IPv6 Introduction; Multicasting

[4 Hrs]

Network Layer, Routing algorithms: Link State, Distance Vector Routing; OSPF, BGP, RIP; Routing Policies

[4 Hrs]

Link Layer: Review of fundamentals of link layer protocols; Ethernet Switches, LANs, Link-Layer Switches, VLANs, Complete tracking of traversal of a packet over internet between two applications

[5 Hrs]

Wireless Networks: Wireless Links and Network Characteristics, Bit Error Rate, SNR; Problems of Wireless Links(Interference, Signal Strength fading, Multipath Propagation, Hidden Terminal problem) , 802.11 (Architecture, MAC protocol, Frame Structure) ,

[8 Hrs]

Text Books

- J.F. Kurose and K. W. Ross, "Computer Networking: A Top-Down Approach Featuring the Internet", Pearson, ISBN-13: 9780201976991
- Behrouz A. Forouzan, Firouz Mosharraf, Computer Networks: A Top-Down Approach, Tata McGraw-Hill Education Pvt. Ltd, ISBN 10: 1259001563 / ISBN 13: 9781259001567
- A S Tanenbaum, "Computer Networks", Pearson Education, ISBN 9788177581652

References

- Larry Peterson Bruce Davie, Computer Networks A Systems Approach, Elsevier, ISBN: 9780123850591
- Kevin R. Fall, W. Richard Stevens, TCP/IP Illustrated, Volume 1: The Protocols, Pearson, ISBN-13: 978-0321336316/ISBN-10: 0321336313
- Behrouz Forouzan, Data Communications and Networking, Tata McGraw-Hill, ISBN-13: 978-0073250328/ISBN-10: 0073250325
- William Stallings, "Data and computer Communication", Pearson Education, ISBN-81-297-0206-1
- Alberto Leon Garcia and Indra Widjaja, "Communication Networks, Fundamental Concepts and Key Architectures", Tata McGraw-Hill, ISBN-10: 007246352X
- Peter Loshin,IPv6 Theory, Protocol, and Practice, Elsevier, ISBN: 9781558608108

(CT- -21006) Computer Networks Laboratory

Teaching Scheme:
Laboratory: 2 Hrs/Week

Examination Scheme:
Term Work: 50 Marks
Oral/End Term Exam: 50 marks

Course Outcomes

Students will be able to:

1. Execute various Networking commands to configure, inspect network on a computing device
2. Demonstrate Socket Programming ability
3. Implement and Execute Packet tracer Programs and analyse all the packets captured
4. Install and configure various Networking applications
5. Write simulations of Networking Protocols using network simulators
6. Design computer networks and sub-networks

List of Assignments

1. Implement a TCP-IP client-server application. The server accepts a string as a request and sends the capitalized version back.
2. Configure network on your computer using DHCP and Static IP. Learn the following networking commands and configuration files: ifconfig, ping,host, traceroute,, telnet, netstat, nslookup, ssh, scp, wget, /etc/resolv.confdf, /etc/hosts, /etc/network/interfaces
3. Install and demonstrate the following networking applications: web server (apache2 or nginx) , ftp server, ssh server, email client (thundereetc. bird, pine, mutt, etc.)
4. Access the website http://go.com (or any website on HTTP, not HTTPS) . Use Wireshark to inspect the flow of packets for a GET request using browser. Identify all packets for this request. Capture the data from all packets and construct the HTML page. Trace the TCP sequence numbers and verify that they match the expected behavior.
5. Send an email using SMTP over the server at new.toad.com (an open SMTP server) .
6. Implement a packet sniffer to capture packets of a specified link layer, network layer, transport layer, or application layer protocol.
7. Write a program to simulate either Go-Back-N or Selective Repeat protocols.
8. Implement a given subnet using ns3 simulator and demonstrate the flow of packets.
9. Design a given subnet using any simulator and demonstrate connectivity among all nodes as specified.
10. Critically analyze the COEP network design and suggest at least one improvement.
11. Create the routing topology specified by instructor and carry out performance evaluation of a specified routing protocol or TCP variant using *ip netns* on Linux.
12. Evaluate the following performance parameters of Linux kernel TCP like receiver advertisement buffer size, initial congestion window, window scaling, using *ip netns.*
13. **Course Mini project:** Implement any one of these: a web server(HTTP Protocol) , email client (IMAP and POP procols) , a bittorrent client, DNS server and client (like nslookup) , etc, simulations of transport layer/network layer protocols; Adding features/bug fixes in apache2/nginx like projects

This list is a guideline. The instructor is expected to improve it continuously.

(ML-21002) Environmental Studies

(Adopted from the 'Ability Enhancement of Compulsory Courses: Environmental Studies' as prescribed by the Expert Committee of University Grants Commission as per directives of Hon'ble Supreme Court)

Teaching scheme
Lectures: 1 hr / week

Evaluation scheme
Periodic Assignments & Tests
Assignment:2 hrs/week

Course Outcomes

At the end of the course, students will demonstrate the ability to:

1. Comprehend Sustainable Development Goals for present generation
2. Appreciate environmental resources, functioning of an ecosystem, significance of biodiversity and environmental challenges
3. Analyze the current status of environment with respect to precautionary mechanisms and control measures
4. Appreciate the role of an engineer for better tomorrow

Course Contents

Unit I

Multidisciplinary nature of environmental studies

Definition, scope and importance. Need for public awareness.

[2 Hrs]

Unit II

Natural Resources: Renewable and non-renewable resources

Natural resources and associated problems .

Forest resources: Use and over-exploitation, deforestation, case studies. Timber extraction, mining, dams and their effects on forest and tribal people.Water resources: Use and over-utilization of surface and ground water,floods, drought, conflicts over water, dams-benefits and problems.Mineral resources: Use and exploitation, environmental effects of extracting and

using mineral resources, case studies.Food resources: World food problems, changes caused by agriculture and overgrazing, effects of modern agriculture, fertilizer-pesticide problems, waterlogging, salinity, case studies.Energy resources: Growing energy needs, renewable and non renewable energy sources, use of alternate energy sources. Case studies.Land resources:

Land as a resource, land degradation, man induced landslides, soil erosion and desertification.

Role of an individual in conservation of natural resources.Equitable use of resources for sustainable lifestyles.

[8 Hrs]

Unit III

Ecosystems

Concept of an ecosystem, Structure and function of an ecosystem, Producers, consumers and decomposers, Energy flow in the ecosystem, Ecological succession, Food chains, food webs and ecological pyramids, Introduction, types, characteristic features, structure and function of the following ecosystem:-Forest ecosystem, Grassland ecosystem, Desert ecosystem, Aquatic ecosystems (ponds, streams, lakes, rivers, oceans, estuaries)

[6 Hrs]

Unit IV

Biodiversity and its conservation

Introduction – Definition: genetic, species and ecosystem diversity, Bio geographical classification of India, Value of biodiversity: consumptive use, productive use, social, ethical, aesthetic and option values, Biodiversity at global, National and local levels, India as a mega-diversity nation, Hot-sports of biodiversity, Threats to biodiversity: habitat loss, poaching of wildlife, man-wildlife conflicts, Endangered and endemic species of India, Conservation of biodiversity: In-situ and Ex-situ conservation of biodiversity.

[8 Hrs]

Unit V

Environmental Pollution

Definition, Cause, effects and control measures of:-Air pollution, Water pollution, Soil pollution, Marine pollution, Noise pollution, Thermal pollution, Nuclear hazards, Solid waste Management: Causes, effects and control measures of urban and industrial wastes, Role of an individual in prevention of pollution, Pollution case studies, Disaster management: floods, earthquake, cyclone and landslides.

[8 Hrs]

Unit VI

Social Issues and the Environment

From Unsustainable to Sustainable development, Urban problems related to energy, Water conservation, rain water harvesting, watershed management, Resettlement and rehabilitation of people; its problems and concerns. CaseStudies, Environmental ethics: Issues and possible solutions, Climate change, global warming, acid rain, ozone layer depletion, nuclearaccidents and holocaust. Case Studies, Wasteland reclamation, Consumerism and waste products. Environment Protection Act, Air (Prevention and Control of Pollution) Act, Water (Prevention and control of Pollution) Act, Wildlife Protection Act, Forest Conservation Act, Issues involved in enforcement of environmental legislation, Public awareness.

[7 Hrs]

Unit VII

Human Population and the Environment

Population growth, variation among nations, Population explosion – Family Welfare Programme, Environment and human health, Human Rights, Value Education, HIV/AIDS, Women and Child Welfare, Role of Information Technology in Environment and human health, Case Studies.

[6 Hrs]

Unit VIII

Field work

Visit to a local area to document environmental assets river/forest/grassland/hill/mountain Visit to a local polluted site-Urban/Rural/Industrial/Agricultural, Study of common plants, insects, birds, Study of simple ecosystems-pond, river, hill slopes, etc.

[5 Hrs]

Reference Books

- Agarwal, K.C. 2001 Environmental Biology, Nidi Publ. Ltd. Bikaner.
- Bharucha Erach, The Biodiversity of India, Mapin Publishing Pvt. Ltd., Ahmedabad – 380 013, India, Email:mapin@icenet.net (R)
- Brunner R.C., 1989, Hazardous Waste Incineration, McGraw Hill Inc. 480p
- Clark R.S., Marine Pollution, Clanderson Press Oxford (TB)
- Cunningham, W.P. Cooper, T.H. Gorhani, E & Hepworth, M.T. 2001,
- Environmental Encyclopedia, Jaico Publ. House, Mumabai, 1196p
- De A.K., Environmental Chemistry, Wiley Eastern Ltd.
- Down to Earth, Centre for Science and Environment (R)
- Gleick, H.P. 1993. Water in crisis, Pacific Institute for Studies in Dev., Environment & Security. Stockholm Env. Institute Oxford Univ. Press. 473p
- Hawkins R.E., Encyclopedia of Indian Natural History, Bombay Natural History Society, Bombay (R)

Humanities and Social Sciences Open Courses-II

(AS (HS)-21005) Industrial Psychology

Teaching Scheme
Lectures: 2 hrs/week

Examination Scheme
Assignment/Test: 40 marks
Final Assessment: 60 marks
Field Visit/Expert Lecture Report: 20 marks
Mini-Project Report: 40 marks

Course Outcomes

1. At the end of the course, student will demonstrate the ability to:
2. determine the psychological factors that influence individual differences at work and appraise the role of research.
3. explain the concepts of motivation and job satisfaction at work and Utilize the elements of organizational culture for enhancing group/team behavior.
4. evaluate the relevance & functioning of leadership & diversity in workforce and acknowledge the multicultural factors influencing workplace behavior.
5. illustrate the process of recruitment & selection and Experiment with the information required to sustain employability.
6. interpret the nuances of Human Factors in Engineering and Analyze its role in their disciplines.
7. measure the behavioral findings from self-lead projects and Propose corrective actions to improve quality of workplace behavior.

Unit 1
Basics of Industrial Psychology (IP)
Difference between IP & Business Programs; Major fields & Employment in IP
Brief History- Scientific Management,Time and Motion Study, Hawthorne Studies,World War I & II
Research in Social Sciences
Individual Differences at Work: Personality, Intelligence, Emotional Intelligence, Creativity & Innovation, Perception & Attitudes

[6 Hrs]

Unit 2
People at Work
Motivation & Job Satisfaction- Employee Predisposition, Expectations, Goals, Incentives& Equity; Job Characteristic Theory (Diagnostic Model)
Understanding Groups & Teams- Group dynamics, Factors affecting Group performance; Understanding work teams, Types of teams, Team development, Issues with teamwork
Leadership (Co-Teaching 4 hrs)- Leader characteristics, Leader & situation, Leader & follower; Specific leadership skills, Introduction to Organizational Development (OD)
Diversity- Multiculturalism- Hofstede's theory, Diversity dynamics

[8 Hrs]

Unit 3
Human Factors Engineering (HFE)
Introduction & Brief History of HFE; Essentials of HFE
Person-Machine Systems- Basic Human Factors: Sensory systems, Perception, Cognition,

Information Processing approach, Memory, Decision Making
Workspace Designs- General Principles, Designing work areas; Machine Displays &Controls; Physical work environment & Anthropometry; Managing workplace strain through Ergonomics (Self-study)
Current trends in HFE- Use of artificial intelligence, cognitive engineering, sociotechnical systems, etc.

[8 Hrs]

Unit 4
Managing People at Work
Job Analysis- Brief Background, Types & Importance; Job description
Recruitment & Selection- Overview, Process, Evaluation
Gearing for Selection- Interviews & Job Search Skills
Performance Appraisal (Co-Teaching 2 hrs): Steps in the Evaluation Process; Appraisal Interview

[6 Hrs]

Text Books

- Aamodt, M.G. (2013). Industrial Psychology. Cengage Learning: Delhi.
- Wickens, C. D.; Lee, J. D., Liu, Y. & Gordon Becker, S. E. (2015). An Introduction to Human Factors Engineering. 2nd Edition. Pearson Education: New Delhi.
- Landy, F. J. & Conte, J. M. (2010). Work in the 21st Century: An Introduction to Industrial and Organizational Psychology. 2nd Edition. Wiley India: New Delhi.

References

- Matthewman, L., Rose, A. & Hetherington, A. (2009). Work Psychology. Oxford University Press: India.
- Schultz, D. & Schultz, S. E. (2013). Psychology and Work Today: An Introduction to Industrial and Organizational Psychology. 7th Edition. Pearson Education: New Delhi.
- Schultz, D. & Schultz, S. E. (2002). Psychology and Work Today. Pearson Education: New Delhi.

(AS (HS)-21006) Personnel Psychology

Teaching scheme
Lectures: 2 hrs /Week

Evaluation scheme
Assignments: 70 marks
End Sem. Exam: 30 marks

Course Outcomes

At the end of the course, student will demonstrate the ability to:

1. acquire organizational concepts and will recognize their own personality attributes suitable for corporate world.
2. realize the importance of motivation and apply motivational principles to their lives
3. experience group dynamics and apply those principles in their lives
4. grasp and apply different techniques to maintain mental health.

Unit 1
Introduction- Understanding own personality and corporate world Basic concepts in Organizational set up and its importance, Know own personality attributes. Preparing for corporate world, work ethics, and self- management

[6 Hrs]

Unit 2
Motivation
Motivational theories for self- motivation and motivating others at work place, Approaches to work

[6 Hrs]

Unit 3
Group dynamics
Group behavior and leadership, Effective group behavior, Leadership and management principles, virtual teams and Performance appraisal

[8 Hrs]

Unit 4
Mental health at work place
Occupational stress and conflict and strategies for its management, Emotional Intelligence, spiritual Intelligence

[6 Hrs]

Text Books

- Khana S.S.- (2016) Organizational Behaviour(Text and Cases) Chand and company Pvt.Ltd.Delhi.
- Rae Andr'e:- (2008) organizational behavior. Dorling Kindersley (India) Pvt. Ltd.
- Wallace H.and Masters L.- (2008) Personality development..Cengage Learning India Pvt. Ltd.

Reference Books

- Robbins S, JudgeA, Vohra N:- (2013)Organizational behavior.(15thed) Pearson Education,Inc.
- Singh Kavita:- (2010) Organizational behavior-Text and cases. Dorling Kindersley

(AS (HS)-21007) Engineering Economics

Teaching Scheme
Lectures: 2 hrs/week

Examination Scheme
Assignment/Test: 40 marks
End Sem. Exam: 60 marks

Course Outcomes

At the end of the course, student will demonstrate the ability to:

1. demonstrate understanding of economic theories and policies.
2. identify economic problems and solve it by applying acquired knowledge, facts and techniques in the available framework.
3. categorize, classify and compare economic situations and draw inferences and conclusions.
4. adapt to changing economic atmosphere and propose alternative solutions to the problems.

Unit 1
Introduction to Economics:
Definitions, basic concepts of economics: Cost, efficiency and scarcity, Opportunity Cost

Types of economics: Micro Economics, Macroeconomics and Managerial Economics.
Difference between micro economics and macroeconomics. Application of Managerial economics

[6 Hrs]

Unit 2
Micro Economics Analysis
Demand Analysis, Supply Analysis, Theories of Utility and Consumers Choice, Cost analysis, Competition and Market Structures. Application of micro economics theories

[8 Hrs]

Unit 3
Macro Economic Analysis
Aggregate Demand and Supply, Economic Growth and Business Cycles, inflation,
Fiscal Policy, National income, theory of Consumption, savings and investments, Commercial and Central banking. Use of macroeconomic theories.

[8 Hrs]

Unit 4
International Economics
Balance of Trade and Balance of Payments, Barriers to Trade, Benefits of Trade/Comparative Advantage, Foreign Currency Markets/Exchange Rates, Monetary, Fiscal and Exchange rate policies, Economic Development.
Application of exchange rate policies

[8 Hrs]

Reference Books

- Macroeconomics: N. Gregory Mankiw, 2018
- Managerial Economics: Economic Tools for Today's Decision Makers: by Paul Keat **(Author),** Philip Young **(Author) 2013**
- Principles Of Macro Economics: Misra and Puri.2009, Himalaya publishing house, New Delhi.
- Modern Microeconomics, A. koutsoyiannis , Macmillan , London
- Microeconomics Robert S. Pindyck and daniel L. rubinfeld:,pearson education Inc. New Delhi
- Micro economics: K. N. Verma

(AS (HS)-21008) Finance for Engineers

Teaching scheme
Lectures:2 hrs / week

Evaluation scheme
Assignments 40 Marks
End Semester 60 Marks

Course Outcomes

At the end of the course, student will demonstrate the ability to:

1. comprehend basics of accounting, cost concepts, will be able to read Financial statements of companies
2. enable them to understand critical financial principles and to enable them to integrate & analyze financial information necessary for Business Decision Making.
3. establish relationship between Risk & Return, time value of money, sources of

finance & working capital
4. appreciate the digital platform of future finance, cryptocurrency, the terms associated with Financial Markets
5. such as Money market, capital market, SEBI & other Regulatory authorities

Unit 1
Introduction to Accounting & Finance
Basic elements of financial accounting, cost concepts, preparation of Profit & Loss Account & Balance Sheet & concept of Budgetary control

[6 Hrs]

Unit 2
Read & interpret Financial Statements
As per Schedule III of Companies Act 2013, Financial statement analysis, concept of cash flow statement

[6 Hrs]

Unit 3
Break-even analysis, Risk & Return relationship, time value of money, sources of finance & working capital

[8 Hrs]

Unit 4
Digital Platform such as Net Banking, Cryptocurrency, Algorithm based stock exchange trading, Basics of Money market, capital market, Commodities market, IPO & Regulatory authorities
****Pedagogy:** Lectures and PPTs, Use of basic Excel tools for preparation of final accounts, Annual Reports of companies.

[4 Hrs]

Reference Books

- Accounting for Managers – C Rama Gopal (2012), Accounting for Management, New Age International Publishers
- Financial Management – Theory and Practice - Prasanna Chandra [Mc Graw Hill] publication

(CT-21015) Software Engineering Mini Project Stage- II

Teaching Scheme
Lectures: 2 Hrs / Week
Laboratory: 2 Hrs/Week

Examination Scheme
Quiz/Assignments: 20 Marks
Theory Exam:40 Marks
Project: 40 Marks (Submission in Stages)

Course Outcomes

Students will be able to:

1. Describe fundamental concepts of system development lifecycle through SDLC.
2. Design user interface prototypes for real world scenarios using appropriate methods of analysis and design.

3. Devise procedure to assure the quality and maintainability of the product before and after deployment.
4. Develop skill to transfer acquired knowledge across a wide range of industrial and commercial domains and have a basis for further studies in software engineering or in computing related industries.
5. Analyze real world scenario and apply tools and techniques to produce application software solutions from informal and semiformal problem specifications.
6. Develop an ability to work in a team by communicating computing ideas effectively in writing a technical report.

Course Contents

Software Development process: Software Engineering basics, Software Crisis and Myths, Software Process and development, Software life cycle and Models, Analysis and comparison of various models, agile process

[6 Hrs]

Requirement Engineering: Requirements Engineering, requirement engineering process, Introduction to Analysis model.

[6 Hrs]

System Architecture and Design Overview, Architecture 4+1 view, architecture styles, Design process, quality concepts, Analysis model to Design Model transformation, Standardisation using UML.

[6 Hrs]

Software Metrics: Introduction to Software Metrics, Size-oriented metrics and function point metrics. Effort and cost estimation techniques -LOC-based and Function-point based measures - The COCOMO model.

[6Hrs]

Testing and Maintenance: Validation and Verification activities, Testing Principles and strategies, Testing levels & types- White Box & Black Box Testing, Maintenance, Types of Maintenance.

[6 Hrs]

Laboratory Mini Project Task

Students will carry out one of the following mini projects:

A) Work on an existing free software/open source project and contribute to it in terms of feature improvements or significant bug fixes. The project will be finalised in consultation with the laboratory instructor. The work can be carried out in teams of any size, however individual evaluation will be carried out on individual basis. The contributions should clearly bring out the following:
   1. Use of version control systems
   2. Contributions of each individual member of the team as seen in version control system
   3. Use of industry standard coding practices
   4. Participation in the software development process of the particular software
   5. Writing test cases and testing the software
   6. Deployment changes, Packaging if needed

7. Acceptance of your contributions by the upstream community.

B) A full-fledged working system in the form of mini project will be implemented following the task list given below. Students in group of two will be working on mini project. After consultation with the course instructor and finalisation of the topic following deliverables are expected under mini-project. Task List for the same is as follows:
   1. Carry out state of art survey, selecting appropriate domain, problem identification, statement formulation based on research problems or real-world problems, industry based problem etc.
   2. Develop workflow graph and carry project estimation, calculation of efforts, project planning (schedule) using automated tools.
   3. Gather requirements and Write the Software Requirement specification (SRS-IEEE specs) document for the project.
   4. Draw different UML diagrams and System architecture for the proposed system. Use different open source tools for design.
   5. Develop Test cases. Propose solution for wrong results in test cases by focusing on regression testing.
   6. Write the constraints, advantages and disadvantages of your project over existing system.
   7. Write the future scope of your project. Develop help manual for maintenance and usability.

Students will be required to submit a technical report written using LaTEX. The technical report will include description of the project/problem, design of the software, description of problems solved and solution design, result analysis of test cases and conclusions. Students will also be required to demonstrate and present their work in a viva-voce.

This list is a guideline. The instructor is expected to improve it continuously.

Text Books

- "Pressman R., "Software Engineering, A Practitioners Approach", 6th Edition, Tata McGraw Hill Publication, 2004, ISBN 007-124083-124083-7.
- "G. Booch, J. Rumbaugh and I. Jacobson. The Unified Modeling Language User Guide, Addison Wesley, 1999.

Reference Books

- "Shari Pfleeger, "Software Engineering", 2nd Edition. Pearson's Education, 2001.
- "Ian Sommerville, "Software Engineering", 6th Edition, Addison-Wesley, 2000
- "Pankaj Jalote, "An Integrated Approach to Software Engineering", Narosa publication house
- "Fred Brooks, "Mythical Manmonths", www.cs.drexel.edu/~yfcai/CS4517/.

Papers

- Fred Brook, "No Silver Bullets", IEEE Software 1987.
- Eric Raymond, "Cathedral and Bazaar ", www.tuxedo.org/~esr/writings.
- David Parnas, "On the Criteria To Be Used in Decomposing Systems into Modules", Communications of the ACM, volume 15, #12, 1972.
- Grady Booch, IEEE software on architecture,IEEE Computer Society.

(IOC-21002) Introduction to Artificial Intelligence

Teaching Scheme:
Lectures: 1 Hr/week
Laboratory: 2 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to

1. Compare AI with human intelligence and traditional information processing and discuss its strength and limitations
2. Apply the basic principles, models and algorithms of AI to recognize, model and solve problems
3. Demonstrate knowledge of basics of the theory and practice of Artificial Intelligence
4. Design and carry out an empirical evaluation of different algorithms on a problem formalisation, and state the conclusions that the evaluation supports.
5. Apply knowledge representation techniques and problem solving strategies to common AI applications.

Course Contents

Introduction: What is AI, History, AI problems, Production Systems, Problem characteristics, Intelligent Agents, Agent Architecture, AI Application (E-Commerce, & Medicine) , AI Representation, Properties of internal representation, Future scope of AI , Issues in design of search algorithms.

[3 Hrs]

Heuristic search techniques: Heuristic search, Hill Climbing, Best first search, mean and end analysis, Constraint Satisfaction, A* and AO* Algorithm**, Knowledge Representation:** Basic concepts, Knowledge representation Paradigms, Structured representation of knowledge, ISA hierarchy

[3 Hrs]

Knowledge Inference: Introduction, Knowledge representation- Production based system, Forward and Backward reasoning, Knowledge representation using non monotonic logic: TMS (Truth maintenance system)

[3 Hrs]

Learning & Planning: What is Learning, Types of Learning (Rote, Direct instruction Analogy, Induction, Deduction) , Planning: Block world, strips, Implementation using goal stack, Non linear planning with goal stacks, Hierarchical planning, Least commitment strategy.

[3 Hrs]

Game playing and Introduction to Machine Learning: Min-max search procedure, Alpha beta cutoffs, waiting for Quiescence, Secondary search, Perception and Action: Perception, Action, Robot Architecture, Machine Learning: Definition of learning systems. Goals and applications of machine learning.

[4 Hrs]

Text Books

- Elaine Rich and Kerin Knight, Artificial Intelligence, 3rd Edition, McGraw Hill. ISBN13: 9780070087705
- Eugene, Charniak, Drew Mcdermott, Introduction to artificial intelligence, Addison-Wesley. ISBN 0-07-052263-4.

Reference Books

- Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 3rd Edition. ISBN 0-13-103805-2.
- Tom Mitchell, Machine Learning, McGraw Hill. *ISBN*-10: 1259096955
- Herbert A. Simon, The Sciences of the Artificial, MIT Press, 3rd Edition, 1998. *ISBN*: 9780262190510.
- George F Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Pearson Edu., 4th Edition. *ISBN*-13: 978-0-321-54589-3.

List of Assignments

1. Implement A* algorithm .
2. Implement AO* algorithm .
3. Implementation of other Searching algorithm.
4. Implementation of Min/MAX search procedure for game Playing .
5. Implementation of variants of Min/ Max search procedure.
6. Implementation of mini Project using the concepts studied in the AI course.

This list is a guideline. The instructor is free to assign new assignments.

Department Elective – I

(CT(DE)-21001) Advanced Data Structures

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Design new operations by using advanced data structures such as priority queues, dictionary structures, and multi-dimensional data structures
2. Analyze the time and space complexity of the operations associated with the advanced data structures and there by appreciate the use of these structures
3. Analyze performance of new data structures
4. Propose new customized structures for efficient dictionary.
5. Apply advanced data structures to solve real life problems.

Course Contents

Review of Basic Concepts: Abstract data types, Data structures, Algorithms, Big Oh, Omega and Theta notations, Solving recurrence equations, Amortized complexity

[4 Hrs]

Priority Queues: Leftist Heap, Skew Heaps, Binomial, Fibonacci and Pairing Heaps, Double ended priority queues

[6 Hrs]

Dictionary Structures: Hash Tables, Universal hash functions, Balanced Binary Search Trees, Splay Trees, 2-3 trees, 2-3-4 trees, Red-black trees, Skip lists, Randomized Dictionary Structures, Treaps.

[6 Hrs]

Multidimensional and Spatial Structures: Interval, Segment, Range, and Priority Search Trees, Quadtrees and Octrees, R-trees.

[8 Hrs]

Miscellaneous Topics: Persistent Data Structures, Cache-Oblivious Data Structures

[8 Hrs]

Applications: IP Router Tables, Data Structures in Web Information Retrieval, Computational Biology, Geographic Information Systems, Computational Geometry: Geometric data structures.

[8 Hrs]

Text Books

- Introduction to Algorithms; 3rd Edition; by by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein; PHI Learning Pvt. Ltd.; ISBN-10: 0262033844; ISBN-13: 978-0262033848
- Advanced Data Structures; by Prof Peter Brass; Cambridge University Press; ISBN-10: 1107439825; ISBN-13: 978-1107439825

Reference Books

- Handbook of Data Structures and Applications; by Dinesh P. Mehta (Editor) , Sartaj Sahni (Editor) ; Chapman and Hall/CRC; ISBN-10: 1584884355;ISBN-13: 978-1584884354

Internet Resources:

- MIT OpenCourseWare
- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-851-advanced-data-structures-spring-2012/index.htm
- COP 5536: Advanced Data Structures: Prof. Sartaj Sahni, University of Florida
- https://www.cise.ufl.edu/~sahni/cop5536/

(CT(DE)-21002) Advanced Data Structures Laboratory

Teaching Scheme:
Laboratory: 2 Hrs/Week

Examination Scheme:
Continuous evaluation: 50 Marks
End Semester Exam: 50 Marks

Course Outcomes

Students will be able to:

1. Implement advanced data structures as abstract data types.
2. Design programs for real life problems using advanced data structures.
3. Choose appropriate data structures for a given problem.

List of Assignments

1. Implement abstract data type for a dictionary as specified by the instructor.
2. Implement ADT for a partition.
3. Implement ADT for priority queue.
4. Implement an online path finding algorithm, which given dynamically changing edge weights suggests all possible paths, including the shortest path between any pair of vertices.
5. Write a program to find the convex hull of a set of points.

This is a suggested list. The instructor is expected to improve it continuously.

(CT(DE)-21003) Advanced Microprocessors

Teaching Scheme
Lectures: 3Hrs/ Week

Examination Scheme
Assignment/Quizzes: 40 marks
End Semester Exam: 60 marks

Course Outcomes

Students will be able to:

1. Analyze the protected mode, privilege levels, various descriptors and support for debugging
2. Justify the need of Inter privilege level access mechanism, Multitasking support and paging facility.
3. Describe interrupt handling mechanism
4. Explain the use of MTRRs and PAT
5. Identify features of MMX technology, SIMD Execution Model and virtualization support

Course Contents

System Architecture Overview: Support for operating-system and system-development software. Multitasking capability, subtasks and modularity in entire system, support offers multiple modes of operation, protection amongst OS and application programs, IA-32 architecture, Intel 64 architecture, System-Level Registers and data structures in protected mode, Global and Local Descriptor Tables

[6 Hrs]

Single-level Task: Protection mechanism and privilege in protected mode, IA-32 architecture, Debugging registers, memory management through segmentation and paging, support registers

[6Hrs]

Multilevel Tasks: Inter privilege level access mechanism call gate. Multitasking support, task switching and task gate in IA-32 processor

[7Hrs]

Interrupts and Memory cache control: Interrupt, exception, faults, traps, interrupt handling in protected mode IDT, interrupt gate, trap gate, interrupt handling in protected mode, V86 mode. Extended features of V86, System Management Mode, Memory cache control: caching terminology, memory types and memory type range registers (MTRRs) ; Page Attribute Table (PAT) , assigning memory types to regions of physical memory based on linear address mappings

[8 Hrs]

MMX and Streaming SIMD extensions: MMX technology, SIMD Execution Model, handling out-of-range conditions, execution environment for the SSE, SSE2, Streaming SIMD Extensions 3(SSE3) , Supplemental Streaming SIMD Extensions 3 (SSSE3) and SSE4

[7Hrs]

VMX Support: Intel Hyper-Threading Technology, Multi-core technology, Intel 64 architecture features, Intel Virtualization Technology

[6Hrs]

Text Books

- Liu & Gibson, Microcomputer Systems, PHI, ISBN: 978-81-203-0409-3
- Barry B. Brey, The INTEL Microprocesoors, PHI, ISBN-81-203-1220-1

Reference Books

- Tom Shanley, Protected Mode Software Architecture MINDSHARE, INC. Addison-Wesley Publishing Company, ISBN: 0-201-55447-X(.pdf)
- Intel® 64 and IA-32 Architectures Software Developer's Volumes 1, 2A, 2B, 3A, 3B (.pdf)

(CT(DE)-21004) Advanced Microprocessors Laboratory

Teaching Scheme
Laboratory: 2Hrs/ Week

Examination Scheme
Continuous evaluation: 50 Marks
Oral: 50 marks

Course Outcomes

Students will be able to:

1. Demonstrate programming in real mode and protected mode.
2. Develop a program using SIMD instructions
3. Illustrate MMX/ SSE/ SSE2 instructions

List of Assignments

1. Write an Assembly program to write (store) a string in Video RAM with help of BIOS Interrupts & display the written string on terminal along with the address of written string.
2. Write an Assembly program to write (store) a string in Video RAM without using BIOS
   Interrupts & display the written string on terminal along with the address of written string.
3. Write an Assembly program to accept any key from user & display the value of key pressed. e.g. input – 'a' desired output -"The key entered is 'a'"
4. Write an Assembly program for boot loader. Display a string "My OS".
5. Write a boot loader which will move from real mode to protected mode. Display message "in real mode ".
6. Write a boot loader which on pressing a key from keyboard transit from real mode to protected mode (Display messages like "in real mode", "press any key to transit into protected mode","in protected mode")
7. Prepare a CASE STUDY on Emulator.
8. Perform matrix operations using MMX/ SSE/ SSE2 instructions.

This list is a guideline. The instructor is expected to improve it continuously.

(CT(DE)-21005) Web Systems and Technologies

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Discuss and critically analyze basic protocols used in World Wide Web.
2. Write client side programming using Javascript, JQuery, Ajax, HTML, CSS
3. Write server side programming using PHP or any server side language.
4. Implement a three tier application using web technologies.
5. Create, publish and test RESTFul web services

Course Contents

Web Essentials: Clients, servers, communication, basic Internet protocols, HTTP Request message, HTTP response message, web clients, generations of web applications, Web server configuration, Debugging tools like Postman

[4 hrs]

Markup languages: HTML: fundamental HTML elements, head, body etc., basic XHTML syntax and semantics, document publishing; CSS: introduction, features, syntax, style properties of text, box, layout, list, table, cursor etc., user defined classes, inheritance.

[4 hrs]

Client-Side Programming: JavaScript: basic syntax, variables and data types, statements, operators, literals, functions; Javascript Objects: properties, references, methods, constructors, arrays, other built-in objects, debugging, host objects, document object model

(DOM) , document tree, DOM event handling, browsers Mobile Applications and Clients, Progressive Web Applications

[8 hrs]

Server-Side Programming: PHP: client request, form data, request headers, server response, HTTP status codes, HTTP response headers, sessions, cookies, URL rewriting, separating programming and presentation, connection to databases.

[8 hrs]

Representing Web Data: XML: namespaces, DOM based XML processing, XSL, X Path, XSLT; AJAX: overview, basics, toolkits, security.

[6 hrs]

Web Services: basic concepts, creating, publishing, testing and describing a web service, WSDL, XML services, communicating object data: SOAP, REST.

[4 hrs]

Javascript Frameworks: Introduction to certain frameworks like JQuery, Nodejs, AngularJS, NodeJS, etc.

[**4 hrs]**

Text Books

- Jeffrey C.Jackson, "Web Technologies: A Computer Science Perspective", Second Edition, Pearson Education, 2007, ISBN 978-0131856035.

Reference Books

- Marty Hall, Larry Brown,"Core Web Programming", Second Edition, Pearson Education, 2001, ISBN 978-0130897930.
- Robert. W. Sebesta, "Programming the World Wide Web", Forth Edition, Pearson Education, 2007, ISBN 978-0321489692.
- H.M. Deitel, P.J. Deitel and A.B. Goldberg, "Internet & World Wide Web How To Program", Third Edition, Pearson Education, 2006, ISBN 978-0131752429.

Online References

- https://www.w3.org/html/
- HTML, The Complete Reference http://www.htmlref.com/
- http://w3schools.org/
- http://php.net/
- https://jquery.com/
- https://developer.mozilla.org/en-US/docs/AJAX
- http://www.tutorialspoint.com/css/

(CT(DE) -21006) Web Systems and Technologies Laboratory

Teaching Scheme:
Laboratory: 2 Hrs/Week

Examination Scheme:
Continuous evaluation: 50 Marks
Practical Exam:50 Marks

Course Outcomes

Students will be able to

1. Demonstrate ability to install, configure a web server
2. Write programs on client side using HTML, Javascript, AJAX, CSS and git
3. Write programs on server side using PHP or any similar technology and demonstrate the ability to install, configure, query a database server and git.
4. Implement a web application using full stack of basic technologies and git.
5. Demonstrate the use of web APIs in developing applications.
6. Demonstrate the use of Javascript frameworks.

Suggested List of Assignments

All the assignments, will be done using git and a remote repo like gitlab/github.

1. Install, configure, compare and discuss features of any open source web server.
2. Install one of these on your computer, configure and setup a website: Wordpress, Drupal or Moodle.
3. Write a HTML page by hand which looks like the homepage of this website: https://www.freecodecamp.org/ or any specified page.
4. Demonstrate use of CSS on any HTML page. Include elements to handle: page resize, changing colour of clicked links
5. Install mysql-server on your laptop and run AQL queries to do the following: create a database, create a table, insert rows in a table, fetch rows from a table, delete a row, update a row.
6. On any HTML page (may the the one you wrote for freecodecamp.org type) , include a link for Login. Write a login page having login/password fields. Write Javascript code to validate the login-id and password for the following: both are properly formed and at least 6 bytes long; the password contains at least one special case, one capital and one numeric characters; convert the password into it's MD5 hash
7. Write a web page which displays a slide show of images. The page should allow changing the timing for the slideshow, automated slideshow, and clicking next/previous buttons.All code should be in one single file. Submit the HTML file and all image files as one single tar.gz, such that on extracting the file (unzipping) and clicking the HTML file the slideshow should start
8. Develop interactive multiple-choice quiz using HTML, JavaScript, AJAX and PHP.
9. Write a web page which reads the address of a location from the user and displays it on google map. OR Write a web page using twitter API, which accepts a twitter user name as input from user, and shows last 10 tweets by that user.
10. Demonstrate checking of form input data using jQuery.
11. **Course Mini Project:** Add features to any existing web application (e.g. Moodle, Wordpress, Drupal, etc. or a project specified by the teacher) .

This list is a guideline. The instructor is expected to improve it continuously.

(CT(DE) -21007) Computer Graphics

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Attain the knowledge about working principles of different Output devices and categories various graphics drawing algorithms.
2. Analytically compare different types of 2D and 3D graphics transformation functions.
3. Design various algorithms for scan conversion and filling of basic objects and their comparative analysis.
4. To implement various algorithms to scan, convert the basic geometrical primitives for clipping.
5. To define the fundamentals of animation, virtual reality and its related technologies.
6. Comprehend viewing techniques (projections of different views of objects along with elimination of invisible components) .

Course Contents

Introduction: Introduction, Application areas of Computer Graphics, overview of graphics systems, graphics environments. Output Primtives: Points and lines, line drawing algorithms, mid-point circle and ellipse algorithms. Polygon and polygon filling algorithms.

[7 Hrs]

Transformations: 2-D Geometric Transformation: Translation, scaling, rotation, reflection and shear transformations, matrix homogeneous coordinates, composite transforms. transformations between coordinate systems.
3-D Geometric Transformation: Translation, rotation, scaling, reflection and shear transformations, composite transformations. Projections.

[7 Hrs]

2-D Viewing and 3-D Viewing: Viewing pipeline, viewing coordinates, view volume. WINDOWING and CLIPPING: Introduction, viewing transformation. clipping, point and line clipping, clipping algorithms, polygon clipping .
Introduction to curves, curve representation (parametric and nonparametric) , general conic curves.

[6 Hrs]

Visible Surface Detection Methods: Introduction, Classification, hidden line removal algorithms, hidden surface removal method. Modelling concepts and techniques. Sweep Representation, Surfaces and Volumes by rotation of curves and surfaces, fractals.

[7 Hrs]

Light and Color Modelling: Introduction Object-Rendering, Light Modeling Techniques, illumination Model, Shading, Flat Shading, Polygon Mesh Shading, Gaurand Shading Model, Phong Shading, Transparency Effect, Shadows, Texture and Object Representation, Color Models.

[7 Hrs]

Segments and Animation: Introduction to segment table, segment functions. Computer animation: Design of animation sequence, animation techniques, key frame systems, motion specifications. Devices for producing animation, computer assisted animation, video formats, method for controlling animation, animation software.

[6 Hrs]

Text Books

- V. K. Pachghare, "Computer Graphics", Third Edition, Laxmi Publications, ISBN-978-81-318-0565-7
- Malay L. Pakhira, " Computer Graphics, Multimedia and Animation", Second edition, PHI, ISBN-978-81-203-4127-2
- Bhattacharya, Samit, "Computer graphics", ISBN: 9780198096191, New Delhi: Oxford University Press, ISBN-978-01-980-9619-1
- Steven Harrington, "Computer Graphics - A Programming approach", Second Edition, TMH, ISBN 0-07-100472-6
- Donald Hearn and M.Pauline Baker, "Computer Graphics C version", Second Edition, Pearson Education, ISBN 81- 7808-794-4

Reference Books

- Dipti P. Mukherjee, "Fundamentals of Computer Graphics and Multimedia", PHI, ISBN-978-81-203-1446-7
- Shah M. B. and B.C. Rana, "Engineering Drawing And Computer Graphics", Pearson, ISBN-978-81-317-5611-9
- David F Rogers, "Procedural elements for Computer Graphics", Second Edition, Tata Mc Graw Hill, ISBN 0-07-047371-4
- F. Hill, "Computer Graphics: Using OpenGL", Second Edition, Pearson Education, ISBN 81-297-0181-2
- J. Foley, V. Dam, S. Feiner, J. Hughes, "Computer Graphics Principles and Practice", Second Edition, Pearson Education, ISBN 81-7808-038-9
- Zhigand xiang, Roy Plastock, "Computer Graphics, Schaum's outlines", Second Edition, Tata Mc- Graw Hill Edition

(CT(DE) -21008) Computer Graphics Laboratory

Teaching Scheme:
Laboratory: 2 Hrs/Week

Examination Scheme:
Continuous evaluation: 50 Marks
End Semester Exam: 50 Marks

Course Outcomes

Students will be able to:

1. Apply the basic concepts of computer graphics and to apply various algorithms for generating and rendering graphical figures
2. Implement the concepts of different type of geometric transformation of objects in 2D and 3D and practical implementation of modeling, rendering, viewing of objects in 2D
3. Execute clipping and filling techniques for modifying an object using illumination and color models techniques to graphics.

4. Demonstrate computer graphics animation using latest animation software with improving self-learning ability.

Suggested List of Assignments

1. Implement line drawing algorithm
2. Implement Mid-point circle drawing algorithm and Explain the utilization of 8-way symmetry technique used for arc drawing
3. Implement polygon drawing and filling functions.
4. Execute 2-D and 3-D transformations (translation, scaling and rotation)
5. Execute graphics tools clipping algorithm
6. Implement graphics tools visible surface detection
7. Execute graphics tools for object rendering with light and color modeling
8. Implement controlling techniques for animation

This list is a guideline. The instructor is expected to improve it continuously.

(CT(DE) -21009) Digital Signal and Image Processing

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Describe the components of DSP system, key DSP concepts and how do they relate to real applications
2. Represent discrete-time signals and systems analytically and visualize them in the time domain and frequency domain
3. Analyze images in the frequency domain using various transforms.
4. Evaluate the techniques for image enhancement and image restoration
5. Explain feature extraction techniques
6. Explain the rapid advances in Machine vision

Course Contents

Introduction: Basic elements of digital signal processing (DSP) system, advantage of digital over analog signal processing, summary of DSP applications and introduction to DSP through these application.
Introduction to signal and system and its properties like linearity, time invariance; Linear convolution, properties of linear convolution, A\D conversion process as sampling, Quantization, encoding, sampling theorem.

[8 Hrs]

Analysis of Signals: Transform domain representation of the signal**,** Fourier transform, Fourier transforms of standard signals, FFT algorithms, direct, divide and conquer approach, radix-2 algorithm

[6 Hrs]

Digital Image Processing fundamentals: Introduction, Image Representation, Monochrome Vision Model, Color Vision Model, Image Sampling and Reconstruction Concepts for monochrome and color Image

Monochrome and Color Image Quantization, Two-dimensional signal processing, Vector-Space Image Representation, Two-Dimensional Fourier Transform, Transform Domain Processing, Fast Fourier Transform Convolution, Discrete Cosine Transform its application in Baseline JPEG

[8Hrs]

Image Improvement Techniques: Image Enhancement, contrast manipulation, Histogram Processing, Image Restoration, Image Denoising

[6 Hrs]

Image Analysis and feature extraction: Edge detection, color edge detection, region and boundary segmentation, object recognition, pattern and pattern classes

[6 Hrs]

Machine Vision: Introduction, definition, Active vision system, Machine vision components, hardware's and algorithms, application of machine vision such as in inspection of parts, identification, industrial robot control, mobile robot application, Competing technologies, CCD line scan and area scan sensor, Videcon and other cameras, Triangulation geometry, resolution passive and active stereo imaging, laser scanner.

[6 Hrs]

Text Books

- J.G. Proakis, D.G. Manolakis, " Digital Signal processing", 4th edition, Pearson Education, ISBN 0131873741
- Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing", 2nd edition, Pearson, ISBN 0-201-18075-8
- Pratt W.K., "Digital Image Processing: PIKS Scientific Inside", 4th edition, ISBN:9780471767770

Reference Books

- Ashok Ambardar, "Digital signal processing: A modern introduction", 1st edition, Cenage learning, ISBN 978-81-315-0837-4
- Milan Sonka, Vaclav Hlavac and Roger Boyle, "Image Processing, Analysis and Machine Vision", Third Edition, Springer, ISBN: 978-1-4899-3216-7
- Forsyth and Ponce, "Computer Vision – A Modern Approach", Second Edition, Prentice Hall, ISBN-13: 978-0-13-608592-8
- Richard Szeliski, "Computer Vision: Algorithms and Applications", Springer, ISBN 978-1-84882-935-0

(CT(DE)-21010) Digital Signal and Image Processing Laboratory

Teaching Scheme:
Laboratory: 2 Hrs/week

Examination Scheme:
Term Work – 50 marks
Oral Exam – 50 marks

Course Outcomes

Students will be able to:

1. Prepare comparative study report of scientific computing tools such as scilab/matlab

2. Implement image processing algorithms
3. Compare image processing algorithms

List of Assignments

1. Generate complex signal and analyze the frequency content using Fourier transform
2. Write a program to perform convolution of two finite length causal sequences
3. Conversion of 24 bit color image to 8 bit, 4 bit, 1 bit image
4. Histogram modification (scaling, offset, amplitude change)
5. Image smoothing, sharpening
6. Edge detection – use of Sobel, Prewitt and Roberts operators
7. Develop color image segmentation algorithm
8. Transform application assignment, baseline JPEG compression technique

This list is a guideline. The instructor is expected to improve it continuously.

(CT(DE) -21011) Parallel Computer Architecture and Programming

Teaching Scheme
Lectures: 3 Hrs / Week

Examination Scheme
Assignment/Quizzes: 40 marks
End Semester Exam: 60 marks

Course Outcomes

Students will be able to:

1. Justify the need of high performance computing.
2. Demonstrate quantitative design principles of parallel computer architectures.
3. Measure and analyze performance through different benchmarks and utilities.
4. Differentiate various parallel computer architectures and programming models.
5. Demonstrate and build a basic cluster setup.

Course Contents

Fundamentals of Performance oriented Architecture Design: Defining Computer Architecture, Trends in Technology, Trends in Cost, Dependability, Measuring, Reporting and Summarizing Performance, Quantitative Principles of Computer Design, Moore's Law, Amdahl's Law, Gustafson's Law, Flynn's Classification of Computer Architectures, Recent Computing Trends, Top 500 Ratings, Fundamentals of Computer Design, Basic and Intermediate concepts of pipelining, Pipeline Hazards, Pipelining Implementation issues.

[5 Hrs]

Instruction-Level Parallelism and its Exploitation: Concepts and Challenges, Basic Compiler Techniques for Exposing ILP, Reducing Branch Costs with Prediction, Scheduling, Overcoming Data Hazards with Dynamic Scheduling, Dynamic Scheduling: Algorithm and Examples, Hardware-Based Speculation, Studies of the Limitations of ILP, Limitations on ILP for Realizable Processors, Hardware versus Software Speculation, ILP Support to Exploit Thread-Level Parallelism.

[8 Hrs]

Data-Level Parallelism in Vector, SIMD: Vector Architecture, Vector Instructions, Vectorizing code, SIMD Instruction Set Extensions for Multimedia, Detecting and Enhancing

Loop-Level Parallelism, Introduction to Graphics Processing Units, Example Heterogeneous Architectures and Case Studies.

[7 Hrs]

Memory Hierarchy Design: Basics of Memory Hierarchy, Cache Performance, Basic Cache Optimizations and numericals, Shared and Private Cache, Virtual Memory, Protection and Examples of Virtual Memory, Memory Technology and Optimizations, The Design of Memory Hierarchies, Study of Memory Hierarchies in different Architectures, Case studies.

[7 Hrs]

Thread-Level Parallelism: Introduction to Shared Memory Architectures, Loosely and Tightly coupled multiprocessors, Centralized Shared-Memory Architectures, Snoopy Bus Cache Coherence, Performance of Shared-Memory Multiprocessors, Distributed Shared Memory and Directory Cache Coherence, Basics of Synchronization, Models of Memory Consistency, Examples of Cache Coherence and Consistency.

[7 Hrs]

Parallel Programming Paradigms: Cluster and Network of Workstations (COW and NOW) , Different ways of building a cluster, Parallel Programming Models: Shared Memory, Message Passing, Data Parallel, MPI/PVM, Parallel Algorithm examples: Matrix Multiplication, Sorting, Introduction to Parallel Programming Languages. Case studies of different cluster/ server architectures. Warehouse-Scale Computers: Architecture, Programming Model and Workloads

[6 Hrs]

Text Books

- "John L Hennessy, David A Patterson, "Computer Architecture: A Quantitative Approach", Fifth Edition, Morgan Kaufmann, 2011, ISBN-13: 978-8178672663
- Kai Hwang, Naresh Jotwani, "Advanced Computer Architecture", Third Edition, Tata McGraw-Hill Edition, 2016, ISBN: 978-9339220921.

Reference Books

- D. E. Culler, J. P. Singh, and A. Gupta, "Parallel Computer Architecture", Second Edition, Morgan Kaufmann, 2017, ISBN: 978-1-4987-7271-6.
- Hesham El-Rewini, Mostafa Abd-El-Barr, "Advanced Computer Architecture and Parallel Processing", Wiley, 2005, ISBN: 9780471467403

(CT(DE)-21012) Parallel Computer Architecture and Programming Laboratory

Teaching Scheme
Laboratory: 2 Hrs / Week

Examination Scheme
Continuous evaluation: 50 Marks
End Semester Oral Exam: 50 Marks

Course Outcomes

Students will be able to:

1. Explain basics of performance analysis of computing systems.
2. Demonstrate performance statistics using perf utility.
3. Implement programs using pthreads and OpenMP constructs.

4. Implement message passing programs in distributed environment.
5. Demonstrate the different steps involved in building of a simple cluster.

Suggested List of Assignments

1. Study of different benchmarks used to evaluate performance of different systems.
2. Performance statistics observation using perf utility.
3. Program to execute matrix multiplication using pthreads.
4. Program to execute matrix multiplication using OpenMP and comparison with pthread program.
5. Program to execute Pi computation and prefix sum using OpenMP.
6. Program to execute section, task and synchronization constructs of OpenMP.
7. Case Study of Cluster building steps - MPI Cluster setup and overview of different routines.
8. Program to implement point to point communication using MPI routines.
9. Program to implement collective communication using MPI routines.
10. Program to implement Map-Reduce parallelism for Warehouse Scale Computer.

Reference Books

- Peter S. Pacheco, "An Introduction to Parallel Programming", Morgan Kaufmann, Morgan Kaufmann, 2011, ISBN: 978-0-12-374260-5.
- Michael Quinn, "Parallel Programming in C with MPI and OpenMP", McGraw-Hill Edition, 2003, ISBN13: 978-0072822564.

This list is a guideline. The instructor is expected to improve it continuously.

(CT(DE)-21013) Computational Geometry

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Demonstrate familiarity with the basic data structures and algorithm design strategies used in computational geometry
2. Demonstrate ability to address implementation issues computational geometry algorithms such as handling degenerate cases
3. Develop and analyze algorithms for simple geometrical problems
4. Apply geometric techniques to real-world problems in areas such as robotics and graphics
5. Carry out time complexity analysis of computational geometry algorithms

Course Contents

Intersections: Introduction to Computational Geometry: The problems studied in Computational Geometry, Applications, terminology, basic Tools. Line Segment Intersection, The Doubly-Connected Edge List, Computing the Overlay of Two Subdivisions, Boolean Operations

[06 Hrs]

Convex Hull Algorithms: Graham's scan, Jarvis Algorithm, a linear expected-time algorithm, applications

[06 Hrs]

Point Location and Triangulation: Point Location and Trapezoidal Maps, Art Gallery Problem, Partitioning a Polygon into Monotone Pieces, Triangulating a Monotone Polygon

[06 Hrs]

Range Searching: 1-Dimensional Range Searching, Kd-Trees, Range Trees, Higher-Dimensional Range Trees, General Sets of Points, Interval Trees, Priority Search Trees, Segment Tree

[08 Hrs]

Arrangements and Duality: Computing the Discrepancy, Duality, Arrangements of Lines, Levels and Discrepancy

[06 Hrs]

Voronoi Diagrams and Delaunay Triangulations: Definition and Basic Properties of Voronoi diagrams, Computing the Voronoi Diagram, Voronoi Diagrams of Line Segments, Farthest-Point Voronoi Diagrams, Triangulations of Planar Point Sets, The Delaunay Triangulation, Computing the Delaunay Triangulation

[08 Hrs]

Text Books

- Mark de Berg, et al., "Computational Geometry: Algorithms & Applications", Springer, 3rd edition, ISBN-13: 978-3540779735
- Joseph O'rourke, "Computational Geometry in C", Cambridge University press, 2nd edition, ISBN-13: 978-0521649766

Reference Books

- Franco P Preparata, et al., "Computational Geometry An Introduction", Springer, ISBN-13: 978-0387961316

Internet Resources:

- NPTEL Course: Computational Geometry
  https://nptel.ac.in/courses/106/102/106102011/

- Lecture Notes by David Mount
  http://www.cs.umd.edu/class/spring2012/cmsc754/lectures.shtml

(CT(DE)-21014) Computational Geometry Laboratory

Teaching Scheme
Laboratory: 2 Hrs / Week

Examination Scheme
Continuous evaluation: 50 Marks
End Semester Exam: 50 Marks

Course Outcomes

Students will be able to:

1. Implement basic and advanced algorithms in the area of computational geometry
2. Demonstrate ability to write code taking care of precision of numbers, boundary conditions, degeneracy etc
3. Verify the implementations for correctness and efficiency by using suitable test data
4. Decide the best possible solution for a given problem under the given constraints
5. Apply the algorithms/concepts studied in the theory course to real life problems

Suggested List of Assignments

1. Area of a polygon
2. Finding closest pair of points
3. Triangulating a polygon
4. Convex hull in two dimensions
5. Convex hull in three dimensions
6. Delaunay Triangulation
7. Segment/ray-segment intersection
8. Segment/ray-triangle intersection
9. Point in polygon
10. Point in polyhedron
11. Intersecting convex polygons
12. Minkowski convolution with a convex polygon
13. Multilink robot arm reachability

This is a suggested list. The instructor is expected to continuously update it.

(CT(DE)-21015) Recent Trends in Computer Networks

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to

1. Explain issues in the design of network topologies and design network systems
2. Analyse different possible solutions for communications at each network layer
3. Simulate working of wired and wireless networks to Explain networking concepts
4. Develop solutions by applying knowledge of mathematics, probability, and statistics to network design problems
5. Explain and Compare various storage and networking technologies
6. Explain the recent advancements in the networking technologies

Course Contents

Internetworking: Routing algorithms evaluations, TCP variants, Quality of Service, Active Queue Management, High-Speed Networks, Performance Modelling and Estimation

[7 Hrs]

IPv6: IPv4 deficiencies, patching work done with IPv4, IPv6 addressing, multicast, Anycast, ICMPv6, Neighbour Discovery, Routing, header compression

[5 Hrs]

Software-Defined Networking and OpenFlow: Centralized and Distributed Control and Data Planes, SDN Controllers, Data Center Concepts, Network Function Virtualization, Mininet, Programming SDNs, Openflow Switch, Wire Protocol, Openstack Neutron plug-in

[6 Hrs]

Ad Hoc Wireless Networks: MAC Protocols for Ad Hoc Wireless Networks, Routing Protocols for Ad Hoc Wireless Networks, Multicast routing in Ad Hoc Wireless Networks, Transport Layer and Security Protocols for Ad Hoc Wireless Networks, Quality of Service in Ad Hoc Wireless Networks.

[8 Hrs]

Storage and Networking: Storage and Networking Concepts, Fiber Channel Internals, Fiber Channel SAN Topologies, Fiber Channel Products, IP SAN Technology, IP SAN Products, Management of SANs, SAN Issues

[7 Hrs]

Advancements in CN: Fundamentals of MPLS (RFC 3031) and GMPLS, SNMP, 5G architecture, Named Data Networks (NDN), Content-Centric Networking (CCN), IoT

[8 Hrs]

Text Books

- Thomas D Nadeau and Ken Grey, Software Defined Networking, O'Reilly, 2013
- https://tools.ietf.org/html/rfc8200
- https://tools.ietf.org/html/rfc3031
- Mani Subramanian, Timothy A. Gonsalves, N. Usha Rani; Network Management: Principles and Practice; Pearson Education India, 2010 References
- William Stallings, High-Speed Networks and Internets, Pearson Education, 2nd Edition, 2002.
- C. Siva Ram Murthy, B.S. Manoj, Ad Hoc Wireless Networks: Architectures and Protocols, Prentice Hall, 2004
- Muthukumaran B, Introduction to High-Performance Networks, Tata Mc Graw Hill, 2008
- Tom Clark, Designing Storage Area Networks, A Practical Reference for Implementing Fibre Channel and IP SANs, Addison-Wesley Professional, 2nd Edition, 2003.
- For 5Garchitecture: https://www.3gpp.org/
- For NDN: https://named-data.net/
- Recent papers and RFCs on advancement in Computer Networks

(CT(DE)-21016) Recent Trends in Computer Networks Laboratory

Teaching Scheme:
Laboratory: 2 Hrs /week

Examination Scheme:
Term Work: 50 marks
Oral Examination: 50 marks

Course Outcomes

1. Relate theory with practice by performing programming assignments
2. Get proficiency in designing network topology solutions
3. Get proficiency in a variety of tools and environments like ns-3, NeST
4. Analyse various networking algorithms and implementation to solve a networking problem
5. Work on the individual assignments to get the
6. Imbibe good programming practices Suggested

Suggested List of Assignments

1. Create a simple dumbbell routing topology setup over the Linux kernel
2. Evaluate the performance of intra-domain routing algorithms using ns-3 or Linux kernel
3. Evaluate the performance of different TCP variants over ns-3 or ip netns
4. Perform a Wireshark packet sniffing experiment.
5. Do the network analysis using NMAP - the Network Mapper.
6. Using Mininet tool for the SDN topology
7. Implement a web proxy that passes requests and data between multiple web clients and web servers.
8. Create a topology using ip netns with the help of IPv6 addresses and perform the operations related to different next headers
9. Evaluate the performance of different Ad Hoc routing protocols e.g., AODV, DSDV, etc,.
10. Setup the NDN testbed (https://named-data.net/)

This list is a guideline. The instructor is expected to improve it continuously.

(CT-21010) Operating Systems

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Write programs to manipulate processes, files, and hardware resources using appropriate system calls.
2. Illustrate the design issues, solutions and complexity of operating system by compiling, modifying an OS kernel, tracing the sequence of activities on processor, data structures of a file system, race conditions, locking mechanisms and storage techniques.
3. Correlate the computer architecture features with operating system design issues.
4. Make design choices for an operating systems with given constraints

5. Discuss suitable changes to existing implementations of operating systems for new features.

Course contents

Introduction and Operating Systems structures: Operating system components, O.S. Services, System Calls, Virtual Machines, Special purpose operating systems, Open-source operating systems, Boot Procedure, Partitions, MBR, GPT, UEFI. Source code of boot loader.; System Services; Linkers and Loaders; Operating system structures; Calling convention and Executable file formats;

[6 Hrs]

Processes and CPU Scheduling: Process concept, interleaved I/O and CPU burst; Process states; Co-operating processes, Thread, Thread libraries, Multithreaded programming, Scheduling, Scheduling criterion, Scheduling algorithms, Multi processor scheduling, Real time scheduling, Interrupts and Interrupt handling. Source code of interrupt handling and scheduler and it's dependence on hardware features.

[6 Hrs]

Process Synchronization: Critical section problem, Hardware support for mutual exclusion, Semaphores, Deadlock-principle, Deadlock detection, prevention and avoidance, Classical problems in concurrent programming: Producer-consumer, Reader-writer with and without bounded buffer. Design of locking primitives like spinlock, semaphore, read-write locks, recursive locks, etc. Source code of spin locks, semaphores, read-write locks.

[8 Hrs]

Inter process Communication: Pipes, Shared memory mechanism, Streams, Asynchronous communication, Signals. A shell program using system calls and pipes.

[4 Hrs]

Memory management: O.S. and hardware interaction, Swapping, Continuous memory management, paging, Segmentation, Virtual Memory Management, Demand Paging, Copy-on-write, Page replacement algorithms, Allocation of frames, Thrashing, Kernel memory management, SVR4 architecture, Unified buffer cache. Case study of x86 architecture and implementation of segmentation and paging in an OS kernel, code of fork(), exec(), sbrk(), etc.

[10 Hrs]

File Management and Storage Structures: File Organization, Concept of files and directories, System calls for file systems, Space allocation issues, Free space management, Data structures like inode and super block, Virtual file system and related object oriented concepts, Disk layout, Formatting, Recovery, NFS, Efficiency and performance, Distributed file systems, Disk Structure, Disk Scheduling, RAID. Case study of a file system like ext2.

[6 Hrs]

Text Books

- Abranhan Silberschatz, Peter B Galvin, Greg Gagne; Operating System Concepts, Wiley India Students Edition, 9th Edition, ISBN: 978-81-265-2051-0

- Andrew S. Tanenbaum; Modern Operating Systems; Prentice Hall of India Publication; 3rd Edition. ISBN: 978-81-203-3904-0
- xv6, a simple, Unix-like teaching operating system, Russ Cox, Frans Kaashoek, Robert Morris; E-book

Reference Books

- Milan Milenkovic; Operating Systems; Tata McGraw Hill; Second Edition. ISBN: 0-07-044700-4
- Maurice J. Bach; The Design of the Unix Opearating System; Prentice Hall of India; ISBN: 978-81-203-0516-8
- Uresh Vahalia; Unix Internals, The New Frontiers; Prentice Hall; ISBN: 0-13-101908-2

(CT-21011) Operating Systems Laboratory

Teaching Scheme:
Laboratory: 2 Hrs/week

Examination Scheme:
Continuous evaluation: 70 Marks
End Semester Exam: 30 Marks

Course Outcomes

Students will be able to

1. Implement features in code of an existing operating system.
2. Demonstrate ability to use existing system programs to inspecting operating system features.
3. Demonstrate the ability to implement system programs.
4. Demonstrate ability to create and solve race conditions using synchronization primitives.
5. Write programs using system calls and IPC.

Suggested List of Assignments

1. Install two operating systems in dual boot mode using virtual machines. One OS will run a GNU/Linux of your choice on it. The other virtual machine will run any non-Linux operating system.
2. Write a minimal version of a shell. The shell should be able to a) execute a program without the complete path name b) handle pipes c) handle redirection d) handle signals
3. Use debugfs tool to locate a file which was recently deleted on an ext2 file system.
4. Write a program, on the lines of debugfs, to browse an ext2 file system and given the complete name name of a file, print it's inode.
5. Download linux kernel source code, compile it and reboot your system with the newly compiled kernel. Add a dummy system call to the Linux kernel. Write a conformance test to test your system call.
6. Implement a list type. Write a code using pthreads for concurrent insertions to the list and demonstrate the problem of race. Then rewrite the program to show how race conditions can be solved by using proper synchronization primitives.
7. Write a program which results in a guaranteed deadlock among it's threads. Kill the threads using operating system's commands to solve the deadlock.

8. Write a program using pthreads to demonstrate the producer consumer problem. Implement appropriate synchronization. Show the different results with and without synchronization.
9. Write a program which acts as a chat application between two users on the same computer, using shared memory.
10. Implement the lseek() system call in xv6 kernel.
11. Rewrite the kernel memory manager in xv6 kernel, that is rewrite the kmalloc() and kfree() using another data structure.
12. **Suggested list of course projects:** Implement mkfs, fsck for ext2 file system; Implement a multithreading library using 1-1, many-many, many-one implementation on Linux; Any of the xv6 based projects: implement demand paging, implement ext2 filesystem, Implement signals, Implement shared memory, Implement Kernel threads, Implement mmap(), etc.

Suggested list of demonstrations by instructor.

1. Trace and explain completely the output of strace on running a "Hello World" C program.
2. Write a program to demonstrate the usage of signals – show how processes can wait for each other, kill each other, stop and continue each other.
3. Demonstrate the changing memory map of a process, by using the contents of the /proc file system, and creative use of malloc() function in the code of the process.
4. Demonstrate that not unmounting a file system results in loss of data. Recover possible data using fsck and restore file system consistency. Analyze the recovered data.

This list is a guideline. The instructor is expected to improve it continuously.

(CT-21014) Design and Analysis of Algorithms

Teaching Scheme:
Lectures: 3 Hrs/week
Tutorial: 1 Hr/week

Examination Scheme:
Assignment/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Derive and solve recurrences describing the performance of recursive algorithms
2. Analyze worst-case running times of algorithms using asymptotic analysis
3. Demonstrate familiarity with standard design techniques and popular algorithms
4. Decide and apply a design technique that would be most suitable to solve a given problem, justify the technique used and analyze the resultant algorithm
5. Formalize and abstract from a given computational task relevant computational problems, reduce problems and argue about complexity classes

Course contents

Introduction: Objectives of time and space analysis of algorithms; Order notations (O, Ω, θ notations) ; (Best average and worst case) time complexity of algorithms such as bubble

sort, selection sort, insertion sort, heap sort etc.; Time complexity of recursive programs using recurrence relations.

[06 Hrs]

Design Techniques-I: Divide and Conquer: Quicksort, Mergesort, Strassen's matrix multiplication, finding convex hull; Greedy Algorithms: Knapsack problem, Job sequencing with deadlines, Optimal merge patterns, Single source shortest paths.

[06 Hrs]

Design Techniques–II: Dynamic Programming: All pairs shortest paths, 0-1 Knapsack, Traveling salesperson problem, Chained matrix multiplication, Longest common subsequence, Bellman-Ford algorithm.

[06 Hrs]

Design Techniques–III: Backtracking: N-queens problem, Hamiltonian cycles, Graph Coloring; Branch-and-Bound: 0/1 Knapsack problem, Traveling salesperson problem

[06 Hrs]

Selected Algorithms from various areas: String Matching: The naïve string-matching algorithm, The Robin-Karp algorithm, The Knuth- Morris-Pratt algorithm; Number -Theoretic algorithms: GCD algorithm, Chinese remainder theorem, Primality testing**;** Network Flow Algorithms: Ford-Fulkerson algorithm, Push-relabel algorithm

[06 Hrs]

Amortised Analysis: Aggregate analysis, accounting method, potential method, dynamic tables; Fibonacci heaps: measurable-heap operations, decreasing a key and deleting a node, bounding the maximum degree; Binomial heaps

[06 Hrs]

Complexity Theory: Lower-bound arguments: Comparison Trees – sorting, Oracles and adversary arguments – merging, finding largest and second largest number in an array; NP-hard and NP-complete problems, proving NP-completeness using reduction technique (e.g. SAT, Independent Set, 3VC, Subset Sum, etc)

[06 Hrs]

Tutorial (Option-1)

Tutorial sessions will consist of solving numerical problems based on the algorithms discussed in theory lectures, discussions around finding complexities of algorithms similar to ones studied in the theory lectures.

Tutorial (Option-2)

1. Solving recurrence relations using different methods such as substitution, recurrence tree, master theorem
2. Numerical problems related to heapsort
3. Tracing of algorithms Quicksort and Mergesort
4. Numerical problems on fractional knapsack, job sequencing with deadlines, Huffman coding, optimal merge pattern

5. Numerical problems on finding shortest paths using Dijkstra's algorithm, finding minimum spanning tree using Prim's and Kruskal's algorithm
6. Numerical problems on dynamic programming – finding optimal way of multiplying a matrix chain, finding longest common subsequence in given two strings
7. Numerical problems on 0/1 knapsack, all pairs shortest paths, Bellman-Ford algorithm
8. Numerical problems on string matching algorithms – Naive, KMP
9. Numerical problems on branch-and-bound technique (0/1 knapsack, travelling salesperson problem)
10. Numerical problems on backtracking technique (n-queens problem, graph coloring, finding Hamiltonian path in a graph, travelling salesperson problem)
11. Numerical problems on network flow algorithms (Ford-Fulkerson and Push-relabel)
12. Finding amortized complexity of a given algorithm
13. Problems related to application of reduction technique to prove a certain problem is NP-complete
14. Recap of the entire course contents

Text Books

- Ellis Horowitz, Sartaj Sahni and Sanguthevar Rajasekaran, "Fundamentals of Computer Algorithms", Universities Press, 2nd edition (2008) , ISBN-13: 978-8173716126
- Thomas Cormen, Charles Leiserson, Ronald Rivest and Cliford Stein, "Introduction to Algorithms", PHI, 3rd edition, ISBN-13: 978-8120340077

Reference Books

- Gilles Brassard and Paul Bratley, "Fundamentals of Algorithmics", PHI, ISBN-13: 978-8120311312
- Jon Kleinberg and Éva Tardos, "Algorithm Design", Pearson Education India, ISBN-13: 978-9332518643

(CT-21012) Data Science

Teaching Scheme	**Examination Scheme**
Lectures: 3 Hrs/ Week	Assignment/Quizzes: 40 marks
Laboratory: 2 Hrs/Week	End Semester Exam: 60 marks

Course Outcomes

Students will be able to:

1. Classify and recognize different types of data
2. Analyze problems in structured framework
3. Determine appropriate data analysis techniques for problem at hand
4. Identify visualization for the data analysis problem
5. Analyze different types of data for inferring meaning

Course Contents

Introduction: Introduction to Data Science, Examples, Data Sources, Challenges, Applications, Introduction to Data Modeling, Statistical Data Modeling, Computational Data Modeling, Statistical limits on data- Bonferroni's principle.

[6 Hrs]

Data gathering and preprocessing: Data gathering: structured and unstructured data, data preprocessing: structured and unstructured data, types, attributes, data cleaning, data integration, data reduction, transformation, discretization.

[8 Hrs]

Exploratory Data Analysis: Descriptive and inferential statistics, Chart types- Single var: Dot plot, Jitter plot, Error bar plot, Box-and-whisker plot, Histogram, Kernel density estimate, Cumulative distribution function, Two variable: Bar chart, Scatter plot, Line plot, Log-log plot, More than two variables: Stacked plots, Parallel coordinate plot, mean, variance.

[6 Hrs]

Data Modeling: What is data model? Function approximation, hypothesis representation, objective / loss function, linera regression, logistic regression, gradient descent.

[8 Hrs]

Similarity Measures, Distance Measures and Frequent Itemsets: Feature extraction - TF, IDF, TF-IDF, Hash functions, Similarity measuring techniques- Shingling, Min-hashing, Locality Sensitive hashing, Distance measures- Triangle Inequality, Euclidean Distance, Cosine Distance, Jaccard Distance, Edit Distance measures, Frequent Itemsets, the Market-Basket Model, Association Rules, A-Priori Algorithm, PCY (Park-Chen-Yu) Algorithm

[6 Hrs]

Data Streams: Stream data model, stream sources, stream queries, issues in stream processing, sampling data in a stream, stream filtering: bloom filter

[6 Hrs]

Text Books

- "Mining of Massive Datasets", Jure Leskovec, Anand Rajaraman, and Jeffery David Ullman, Cambridge University Press, 2 edition (13 November 2014) , ISBN-10: 1107077230, ISBN-13: 978-1107077232
- "Data Mining: Concepts and Techniques", Jiawei Han, Micheline Kamber, 3rd Edition, Morgan Kaufmann, ISBN-13: 978-9380931913

Reference Books

- "Foundations of Data Science", Avrim Blum, John Hopcroft, and Ravindran Kannan, Hindustan Book Agency, (online free version) January 2020, ISBN-10: 9386279800
- "Introduction to Machine Learning", Ethem Alpaydin, PHI Learning Pvt Ltd, Third edition, 2016, ISBN-978-81-203-5078-6

(CT-21013) Data Science Laboratory

Teaching Scheme
Laboratory: 2 Hrs/ Week

Examination Scheme
Continuous evaluation: 50 Marks
Mini Project: 25 marks
End Semester Exam: 25 Marks

Course Outcomes

Students will be able to:

1. Describe data analysis problem in structured framework
2. Determine appropriate data analysis techniques for problem at hand
3. Identify visualization for the data analysis problem
4. Analyze different types of data for inferring meaning
5. Apply learnt techniques to solve real life problems

Suggested List of Assignments

1. Choose a data set and perform following steps for the selected problem statement. (Mini project)
   a. Define problem statement
   b. Data gathering
   c. Data preprocessing
   d. Data exploration and analysis
   e. Data modeling
   f. Model evaluation
   Some samples are listed as follows:
      i. Exacerbations in Chronic Obstructive Pulmonary Disease: Identification and Prediction Using a Digital Health System
      ii. Ad2Vec: Similar Listings Recommender for Marketplace
      iii. Clothes Classification with the DeepFashion Dataset and Fastai
      iv. recommendation lab applications for some select commercial applications
2. Select existing achieved tasks in open online platforms or social competitions and demonstrate the competition result comparisons. (minimum 3 tasks) .

This list is a guideline. The instructor is expected to improve it continuously.

Minor in Computer Engineering

(CT(MI)-21001) Data Structures, Files and Algorithms

Teaching Scheme:
Lectures: 3 Hrs/week

Examination Scheme:
Programming Tasks/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1 Write neat code by selecting appropriate data structure and demonstrate a working solution for a given problem.
2 Demonstrate the ability to implement different data structures with variety of implementations.

3 Analyze and compare given algorithms for time and space complexity.
4 Handle all possible cases in designing an algorithm.
5 Demonstrate the ability to write modular and re-usable code.

Course Contents

Introduction: Concept of Data types and Abstract Data types; Characteristics of an algorithm; Analyzing programs; Frequency count; Time and space complexity; Big 'O' and 'Ω" notation; Best, average and worst cases; Programming language provided data types, operations on various data types; Dangling pointers and garbage memory

[4 Hrs]

Arrays, Searching and Sorting: Searching: linear and binary search algorithm; Hashing: hashing functions, chaining, overflow handling with and without chaining, open addressing: linear, quadratic probing; Sorting: bubble sort, selection sort, quick sort, merge sort, insertion sort. Time complexity analysis of searching and sorting techniques.

[8 Hrs]

Files: Files handling: various library functions for handling files; system call interface and library interface; Different file formats like csv, pdf, odt, etc; Text and binary files; Programs to copy, concatenate, rename files; Programs to handle existing file types; arguments to main() ;

[6 Hrs]

Stacks and Queues: Stack and queue as ADT; Operations on stack and queue; Implementations using arrays and dynamic memory allocation; Application of stack for expression evaluation, expression conversion; Recursion and stacks; Problems like maze and knight's tour.

[6 Hrs]

Lists: List as ADT; Concept of linked organization of data against linked list; Singly linked list, doubly linked list, circular linked list; Representation & manipulations of polynomials/sets using linked lists; Dynamic memory management; Representation of sparse matrix; Addition and transpose of sparse matrix; Polynomials; Representing Numbers

[8 Hrs]

Trees: Basic terminology; Binary trees and its representation; Binary tree traversals (recursive and non recursive) and various operations; Insertion and deletion of nodes in binary search tree; Applications of trees.

[8 Hrs]

Text Books

- E. Horowitz, S. Sahni, S.Anderson-freed, "Fundamentals of Data Structures in C", Second Edition, University Press, ISBN 978-81-7371-605-8
- B. Kernighan, D. Ritchie, "The C Programming Language", Prentice Hall of India, Second Edition, ISBN 81-203-0596-5
- Y. Langsam, M. Augenstin and A. Tannenbaum, "Data Structures using C", Pearson Education Asia, First Edition, 2002, ISBN 978-81-317-0229-1

Reference Books

- Ellis Horowitz, S. Sahni, D. Mehta "Fundamentals of Data Structures in C++", Galgotia Book Source, New Delhi 1995 ISBN 16782928
- Jean-Paul Tremblay, Paul. G. Soresan, "An introduction to data structures with Applications", Tata Mc-Graw Hill International Editions, 2nd edition 1984, ISBN-0-07-462471-7

(CT(MI)-21002) Object Oriented Programming and Design

Teaching Scheme:
Lectures: 3 Hr/week

Examination Scheme:
Programming Tasks/Quizzes – 40 marks
End Sem Exam - 60 marks

Course Outcomes

Students will be able to:

1. Design a class hierarchy using object oriented thinking for a given problem.
2. Create object oriented application code for a given problem.
3. Write small pieces of code demonstrating various object oriented programming concepts.
4. Compare, annotate, and comment on various object oriented programming concepts.
5. Demonstrate ability to write concurrent programs for given problems.

Course Contents

Introduction: Various programming paradigms: Procedural, object-oriented, logic and functional, concurrent programming. Classes, Objects, Methods; Data types provided by OO languages; Input/Output mechanisms; Abstract Data Types; Private, public, protected members; **[8 Hrs]**

Abstraction Mechanisms: Encapsulation; Constructors, Destructors; Polymorphism; Access specification

[6 Hrs]

Inheritance: Multiple Inheritance; Class hierarchies; Virtual Functions;

[8 Hrs]

Standard Libraries: Templates; Generic Programming; Packages; Interfaces. Iterators; Containers.

[8 Hrs]

Exception Handling & File I/O: Exception handling, Exception types, file I/O.

[6 Hrs]

Multi-Threaded Programming: Concurrent Programming, Basic Concepts of Concurrent Programming, Threads. Design: Unified Modelling language; use case diagrams; Class Diagrams;

[6 Hrs]

Text Books

- Cay S Horstmann and Gary Cornell, Core Java Vol-1 and Vol-2, 9th Edition, Pearson Education India, ISBN-10: 9332518904 and 9332518890
- Bjarne Stroustrup, The C++ Programming Language, 3th Edition, Pearson Education, ISBN-10: 8131705218
- M. Ben Ari, "Principles of Concurrent Programming, 1989

Reference Books

- Herbert Schilt, "JAVA Complete Reference", 7th Edition, Tata McGraw Hill, ISBN: 9780070636774
- Sharon Zakhour, Scott Hommel, Jacob Royal, Isaac Rabinovitch, Tom Risser, Mark Hoeber, "The Java Tutorial,"Addison Wesley Professional, 2006, Print ISBN-10: 0-321-33420-5
- Eckel B., "Thinking in Java", 3rd Edition, Pearson Education, 2012
- E. Balagurusamy, Object Oriented Programming with C++, 6th Edition, McGraw Hill, ISBN-10: 125902993X

Honours in Data Science

(CT(HO)-21001) Making Sense of Data

Teaching Scheme
Lectures: 3 hrs/week

Examination Scheme
T1, T2 – 20 marks each
End-Sem Exam – 60 marks

Course Outcomes

Students will be able to:

1. Identify different types of data,
2. Analyze numerous amount of data.
3. Apply pre-processing on data
4. Understand the different data analytics tools
5. Visualize the data

Course Contents

Introduction:
Overview, Sources of Data, Process for Making Sense of Data. Describing Data: Observations and Variables, Types of Variables, types of data, Data Distributions: Discrete Distributions such as Binomial, Poisson, Geometric etc.; Continuous Distributions such as Exponential, Normal etc.; Expectation: Moments; Central Limit theorem and its significance; Some sampling distributions like chi-square, t, F

[06Hrs]

Statistical Inference: Estimation - Introduction, classical methods of estimation, single sample: estimating the mean and variance, two samples: estimating the difference between two means and ratio of two variances; Tests of hypotheses - Introduction, testing a statistical hypothesis, tests on single sample and two samples concerning means and variances; ANOVA - One–way, Two–way with/without interactions.

[08Hrs]

Data Analytics Tools
Data analytics using Python: Statistical Procedures, NumPy, Pandas, SciPy, Matplotlib

[08Hrs]

Data Pre-Processing
Understanding the Data, Dealing with Missing Values, Data Formatting, Data Normalization, Data Binning, Importing and Exporting data in Python, Turning categorical variables into quantitative variables in Python, Accessing Databases with Python.

[06 Hrs]

Data Visualization
Graphic representation of data, Characteristics and charts for effective graphical displays, Chart types- Single var: Dot plot, Jitter plot, Error bar plot, Box-and-whisker plot, Histogram, Two variable: Bar chart, Scatter plot, Line plot, Log-log plot, More than two variables: Stacked plots, Parallel coordinate plot.

[06Hrs]

Text Books

2. Glenn J. Myatt, Wayne P. Johnson, Making Sense of Data I: A Practical Guide to Exploratory Data Analysis and Data Mining, Wiley 2009.
3. Glenn J. Myatt, Wayne P. Johnson, Making Sense Of Data II A Practical Guide To Data Visualization, Advanced Data Mining Methods, And Applications,Wiley 2009

References

1. H. Davenport, Jeanne G. Harris and Robert Morison, "Analytics at Work: Smarter Decisions, Better Results", Harvard Business Press, 2010
2. Anil Maheshwari, "Data Analytics made accessible," Amazon Digital Publication, 2014.

(CT(HO)-21003) Big Data Analytics

Teaching Scheme
Lectures: 3 Hrs/ Week

Examination Scheme
Assignment/Quizzes: 40 marks
End Semester Exam: 60 marks

Course Outcomes

Students will be able to:

1. Explain the need of Big Data, challenges and technology stack of big data.
2. Explain and work on Hadoop Framework and ecosystems.
3. Explain and Analyze Big Data using Map Reduce programming framework in both Hadoop and Apache Spark.
4. Analyze large datasets using Apache Pig.
5. Perform Data Querying and Analysis on large datasets using Apache Hive.

Course Contents

Introduction to Big Data: Types of Digital Data, Overview of Big Data Analytics, Evolution of Big Data, Characteristics of Big Data - Volume, Variety, Velocity, Veracity, Valence, Value, Applications of Big Data, Challenges with Big data, Introduction to Enabling Technologies for Big Data.

[5 Hrs]

Big Data Technology Stack: Introduction to Big Data Technology Stack, Overview of Big Data distribution packages, Introduction to Big Data Platforms, Big Data Storage Platforms for Large Scale Data Storage, CAP Theorem, Eventual Consistency, Consistency Trade-Offs, ACID and BASE, Overview of Zookeeper and Paxos, Cassandra.

[7 Hrs]

Apache Hadoop: History of Hadoop, Overview Apache Hadoop and Apache Spark, Hadoop Storage Framework – Hadoop Distributed File System (HDFS) , HDFS Architecture, HDFS Concept, Hadoop Data Processing Framework – Map Reduce, Anatomy of a Map Reduce Job Run, Failures, Job Scheduling, Shuffle and Sort, Task Execution, Map Reduce Types and Formats, Map Reduce Features, MapReduce Programming Model with Spark.

[8 Hrs]

Introduction to Big Data Streaming: Big Data Pipelines for Real-Time computing, Introduction to Apache Spark Streaming, Kafka, Streaming Ecosystem, Spark Components, Resilient Distributed Dataset and data frames.

[7Hrs]

Apache PIG: What is ETL, Introduction to Apache PIG, Execution Modes of PIG, Comparison of PIG with SQL and No-SQL Databases, PIG Data Types, Data Models in PIG, Grunt, PIG Latin, Overview of PIG User Defined Functions

[7 Hrs]

Apache HIVE: Introduction to Apache Hive, Hive Architecture and components, Hive Metastore, Comparison with Traditional Databases, HiveQL, Hive Partition, Hive Bucketing, Tables(Managed and External)

[6 Hrs]

Text Books

- "Big Data Analytics", Seema Acharya, SubhasiniChellappan, Second Edition, 2019, Wiley India Pvt.Ltd, ISBN 978-81-2657-951-8.
- "Hadoop: The Definitive Guide", Tom White, Fourth Edition, 2015, O'Reilly, ISBN 978-93-5213-067-2.

Reference Books

- "Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics", Bill Franks, First Edition,2012, John Wiley & sons,ISBN: 978-1-118-20878-6
- "Mining of Massive Datasets", Jure Leskovec, Anand Rajaraman and Jeffrey David Ulman, Second Edition 2016, Dreamtech Press, ISBN - 978-13-1663-849-1
- "Analytics in a Big Data World: The Essential Guide to Data Science and its Applications", Bart Baesens, First Edition 2014, Wiley, ISBN: 1118892704
- "Big Data Glossary", Pete Warden, First Edition 2011, O'Reily, ISBN
- "Harness the Power of Big Data The IBM Big Data Platform ",Paul Zikopoulos ,Dirk DeRoos , Krishnan Parasuraman , Thomas Deutsch , James Giles , David Corigan , Annotated Edition 2012, Tata McGraw Hill Publications, ISBN 978-0071808170

- "Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Businesses", Michael Mineli, Michele Chambers, Ambiga Dhiraj, First Edition 2013, Wiley Publications, ISBN 978-1118147603
- "BigDataAnalytics: Disruptive Technologies for Changing the Game", ArvindSathi, MC Press, 2012, ISBN 1583473807

Honours in Information Security

(CT(HO)-21002) Fundamentals in Information and Coding Theory

Teaching Scheme
Lectures: 3 Hrs / Week

Examination Scheme
Assignment/Quizzes: 40 marks
End Semester Exam: 60 marks

Course Outcomes

Students will be able to:

1. Apply information theory and linear algebra in source coding and channel coding
2. Comprehend various error control encoding and decoding techniques
3. Analyze the performance of error control codes
4. Apply convolution codes for performance analysis & cyclic codes for error detection and correction
5. Demonstrate applicability of Shannon's security, wiretap model, degraded wiretap model and calculation of secrecy capacity.

Course Contents

Source Coding
Introduction to Information Theory, Uncertainty and Information, Average Mutual Information and Entropy, Information Measures for Continuous Random Variables, Relative Entropy, Source Coding Theorem, Huffman Coding, Shannon-Fano-Elias Coding, Arithmetic Coding, The Lempel-Ziv Algorithm, Run Length Encoding, Rate Distortion Function, Entropy Rate of a Stochastic Process.

[6 Hrs]

Compression techniques
Introduction to Image Compression, The JPEG Standard for Lossless Compression, The JPEG Standard for Lossy Compression, Video Compression Standards, Project Ideas based on the above topics

[6Hrs]

Channel Capacity and Coding
Channel Models, Channel Capacity, Channel Coding, Information Capacity Theorem, Parallel Gaussian Channels, The Shannon Limit, Channel Capacity for MIMO Systems, Capacity Region for Multiple Access Channels, Random Selection of Codes

[6Hrs]

Error Control Coding
Linear Block Codes for Error Correction, Introduction to Error Correcting Codes, Matrix Description of Linear Block, Equivalent Codes, Parity Check Matrix, Decoding of a Linear

Block Code, Syndrome Decoding, Error Probability after Coding (Probability of Error Correction) , Hamming Codes, Low Density Parity Check (LDPC) Codes, Optimal Linear Codes, Maximum Distance Separable (MDS) Codes, Bounds on Minimum Distance, Space Time Block Codes

[6 Hrs]

Cyclic Codes

Introduction to Cyclic Codes, Polynomials, The Division Algorithm for Polynomials, A Method for Generating Cyclic Codes, Matrix Description of Cyclic Codes, Quasi-Cyclic Codes and Shortened Cyclic Codes, Burst Error Correction, Fire Codes, Golay Codes, Cyclic Redundancy Check (CRC) Codes, Circuit Implementation of Cyclic Codes

[6 Hrs]

Convolutional Codes

Tree Codes and Trellis Codes, Polynomial Description of Convolutional Codes (Analytical Representation) , Viterbi Decoding of Convolutional Codes, Turbo Codes, Turbo Decoding

[5 Hrs]

Physical Layer Security

Introduction to Physical Layer Security, Shannon's Notion of Security, The Wiretap Model, Learning Review, The Gaussian Wiretap Model, Secrecy Capacity in Wireless Channels, Cooperative Jamming

[5 Hrs]

Text Books

- "Information Theory, Coding & Cryptography", Ranjan Bose, Third Edition, 2016, McGraw Hill India, ISBN-13: 978-9385880568.
- "Elements of information theory" Thomas M. Cover, and Joy A. Thomas, John Wiley & Sons, 2012.

Reference Books

- "Algebraic Codes for Data Transmission", R. E. Blahut, Cambridge University Press Cambridge, UK, 2003
- "Error Control Coding - Fundamentals and Applications", S. Lin and D.J. Costello Second Edition, Pearson Education Inc., NJ., USA, 2004.
- "Algebraic Coding Theory: Revised Edition", Elwyn R. Berlekamp, World Scientific, 2015.

(CT(HO)-21004) Ethical Hacking

Teaching Scheme

Lectures: 3 Hrs / Week

Examination Scheme

Assignment/Quizzes: 40 marks

End Semester Exam: 60 marks

Course Outcomes

Students will be able to:

1. Identify legal and ethical issues related to vulnerability and penetration testing.
2. Report on the strengths and vulnerabilities of the tested network.

3. Exploit the vulnerabilities related to computer system and networks using state of the art tools and technologies
4. Execute a penetration test using standard hacking tools in an ethical manner
5. Evaluate best practices in security concepts to maintain confidentiality, integrity and availability of computer systems

Course Contents

Introduction to Hacking

Ethical hacking process, Hackers behaviour & mindset, Maintaining Anonymity, Hacking Methodology, Information Gathering, Active and Passive Sniffing, Physical security vulnerabilities and countermeasures. Internal and External testing. Preparation of Ethical Hacking and Penetration Test Reports and Documents.

[10 Hrs]

Social Engineering and Attacks

Social Engineering attacks and countermeasures. Password attacks, Privilege Escalation and Executing Applications, Network Infrastructure Vulnerabilities, IP spoofing, DNS spoofing, Wireless Hacking: Wireless footprint, Wireless scanning and enumeration, Gaining access (hacking 802.11) , WEP, WPA, WPA2.

[10 Hrs]

Application Vulnerabilities and Attacks

DoS attacks. Web server and application vulnerabilities, SQL injection attacks, Vulnerability Analysis and Reverse Engineering, Buffer overflow attacks. Client-side browser exploits, Exploiting Windows Access Control Model for Local Elevation Privilege. Exploiting vulnerabilities in Mobile Application

[10 Hrs]

Introduction to Metasploit

Metasploit framework, Metasploit Console, Payloads, Metrpreter, Introduction to Armitage, Installing and using Kali Linux Distribution, Introduction to penetration testing tools in Kali Linux. Case Studies of recent vulnerabilities and attacks.

[10 Hrs]

Text Books

- "Ethical Hacking and Penetration Testing Guide", Baloch R, CRC Press, 2015.
- " Hacking for Dummies " Beaver, K., Third edition John Wiley & Sons, 2013.

Reference Books

- "Network Forensics Tracking Hackers through Cyberspace ", Davidoff, S. and Ham, J., Prentice Hall, 2012.
- "Computer Forensics JumpStart", Michael G. Solomon, K Rudolph, Ed Tittel, Broom N., and Barrett, D, Willey Publishing Inc, 2011

Started on	Monday, 21 February 2022, 7:01:47 PM
State	Finished
Completed on	Monday, 21 February 2022, 7:56:47 PM
Time taken	55 mins
Grade	**7.90** out of 10.00 (**79**%)

Question **1**

Complete

Mark 0.50 out of 1.00

Which of the following is not a task of the code of swtch() function

- [ ] a. Jump to next context EIP
- [ ] b. Switch stacks
- [ ] c. Load the new context
- [ ] d. Save the old context
- [x] e. Change the kernel stack location
- [ ] f. Save the return value of the old context code

The correct answers are: Save the return value of the old context code, Change the kernel stack location

Question **2**

Complete

Mark 1.00 out of 1.00

Select the odd one out

- ( ) a. Kernel stack of scheduler to kernel stack of new process
- (•) b. Kernel stack of new process to kernel stack of scheduler
- ( ) c. Kernel stack of new process to Process stack of new process
- ( ) d. Kernel stack of running process to kernel stack of scheduler
- ( ) e. Process stack of running process to kernel stack of running process

The correct answer is: Kernel stack of new process to kernel stack of scheduler

Question **3**

Complete

Mark 0.00 out of 1.00

A process blocks itself means

- ( ) a. The kernel code of system call calls scheduler
- (•) b. The kernel code of an interrupt handler, moves the process to a waiting queue and calls scheduler
- ( ) c. The kernel code of system call, called by the process, moves the process to a waiting queue and calls scheduler
- ( ) d. The application code calls the scheduler

The correct answer is: The kernel code of system call, called by the process, moves the process to a waiting queue and calls scheduler

Question **4**

Complete

Mark 1.00 out of 1.00

The trapframe, in xv6, is built by the

- a. vectors.S, trapasm.S
- b. hardware, vectors.S, trapasm.S
- c. hardware, vectors.S
- d. hardware, vectors.S, trapasm.S, trap()
- e. hardware, trapasm.S

The correct answer is: hardware, vectors.S, trapasm.S

Question **5**

Complete

Mark 1.00 out of 1.00

What will be the output of this program

```
int main() {
  int fd;
  printf("%d ",   open("/etc/passwd", O_RDONLY));
  close(1);
  fd = printf("%d ", open("/etc/passwd", O_RDONLY));
  close(fd);
   fd = printf("%d ", open("/etc/passwd", O_RDONLY));
}
```

- a. 3 3 3
- b. 2 2 2
- c. 3 4 5
- d. 3 1 1
- e. 1 1 1
- f. 3 1 2

The correct answer is: 3 1 1

Question **6**

Complete

Mark 0.50 out of 0.50

Match the File descriptors to their meaning

2	Standard error
1	Standard output
0	Standard Input

The correct answer is: 2 → Standard error, 1 → Standard output, 0 → Standard Input

Question **7**

Complete

Mark 0.00 out of 0.50

Which of the following state transitions are not possible?

- [x] a. Ready -> Terminated
- [x] b. Waiting -> Terminated
- [x] c. Ready -> Waiting
- [ ] d. Running -> Waiting

The correct answers are: Ready -> Terminated, Waiting -> Terminated, Ready -> Waiting

Question **8**

Complete

Mark 0.50 out of 0.50

Match the MACRO with it's meaning

PHYSTOP	224 MB
KERNBASE	2 GB
KERNLINK	2.224 GB

The correct answer is: PHYSTOP → 224 MB, KERNBASE → 2 GB, KERNLINK → 2.224 GB

Question **9**

Complete

Mark 1.00 out of 1.00

Arrange in correct order, the files involved in execution of system call

trap.c	4
vectors.S	2
trapasm.S	3
usys.S	1

The correct answer is: trap.c → 4, vectors.S → 2, trapasm.S → 3, usys.S → 1

Question **10**

Complete

Mark 0.90 out of 1.00

Match the elements of C program to their place in memory

Code of main()	Code
Local Static variables	Data
Local Variables	Stack
Arguments	Stack
Malloced Memory	Heap
#define MACROS	No Memory needed
Global variables	Data
Function code	Code
#include files	No Memory needed
Global Static variables	Data

The correct answer is: Code of main() → Code, Local Static variables → Data, Local Variables → Stack, Arguments → Stack, Malloced Memory → Heap, #define MACROS → No Memory needed, Global variables → Data, Function code → Code, #include files → No memory needed, Global Static variables → Data

Question **11**

Complete

Mark 0.50 out of 0.50

Match the names of PCB structures with kernel

linux	struct task_struct
xv6	struct proc

The correct answer is: linux → struct task_struct, xv6 → struct proc

Question **12**

Complete

Mark 1.00 out of 1.00

The "push 0" in vectors.S is

- a. Place for the error number value
- b. To be filled in as the return value of the system call
- c. A placeholder to match the size of struct trapframe
- d. To indicate that it's a system call and not a hardware interrupt

The correct answer is: Place for the error number value

◄ Description of some possible course mini projects

Jump to...

(Code) mmap related programs ►

Started on	Saturday, 26 February 2022, 5:19:53 PM
State	Finished
Completed on	Saturday, 26 February 2022, 6:34:54 PM
Time taken	1 hour 15 mins
Grade	**11.66** out of 15.00 (**78**%)

Question **1**

Complete

Mark 1.00 out of 1.00

Given below is the output of the command "ps -eo min_flt,maj_flt,cmd" on a Linux Desktop system. Select the statements that are consistent with the output

```
626729 482768 /usr/lib/firefox/firefox -contentproc -parentBuildID 20220202182137 -prefsLen 9256 -prefMapSize 264738 -appDir /usr/lib/firefox/browser 6094 true rdd
2167 687 /usr/sbin/apache2 -k start
1265185 222 /usr/bin/gnome-shell
102648 111 /usr/sbin/mysqld
9813 0 bash
15497 370 /usr/bin/gedit --gapplication-service
```

- [x] a. Firefox has likely been running for a large amount of time
- [x] b. The bash shell is mostly busy doing work within a particular locality
- [x] c. All of the processes here exihibit some good locality of reference
- [x] d. Apache web-server has not been doing much work

The correct answers are: Firefox has likely been running for a large amount of time, Apache web-server has not been doing much work, The bash shell is mostly busy doing work within a particular locality, All of the processes here exihibit some good locality of reference

Question **2**

Complete

Mark 0.67 out of 1.00

Shared memory is possible with which of the following memory management schemes ?

Select one or more:

- [x] a. paging
- [ ] b. segmentation
- [ ] c. continuous memory management
- [x] d. demand paging

The correct answers are: paging, segmentation, demand paging

Question **3**

Complete

Mark 0.71 out of 1.00

Compare paging with demand paging and select the correct statements.

Select one or more:

- [ ] a. Demand paging requires additional hardware support, compared to paging.
- [ ] b. Paging requires NO hardware support in CPU
- [x] c. Calculations of number of bits for page number and offset are same in paging and demand paging.
- [ ] d. TLB hit ration has zero impact in effective memory access time in demand paging.
- [x] e. Paging requires some hardware support in CPU
- [x] f. With demand paging, it's possible to have user programs bigger than physical memory.
- [ ] g. With paging, it's possible to have user programs bigger than physical memory.
- [ ] h. Both demand paging and paging support shared memory pages.
- [x] i. Demand paging always increases effective memory access time.
- [x] j. The meaning of valid-invalid bit in page table is different in paging and demand-paging.

The correct answers are: Demand paging requires additional hardware support, compared to paging., Both demand paging and paging support shared memory pages., With demand paging, it's possible to have user programs bigger than physical memory., Demand paging always increases effective memory access time., Paging requires some hardware support in CPU, Calculations of number of bits for page number and offset are same in paging and demand paging., The meaning of valid-invalid bit in page table is different in paging and demand-paging.

Question **4**

Complete

Mark 0.75 out of 1.00

which of the following, do you think, are valid concerns for making the kernel pageable?

- [x] a. The kernel must have some dedicated frames for it's own work
- [ ] b. The disk driver and disk interrupt handler should not be pageable
- [ ] c. No part of kernel code should be pageable.
- [ ] d. No data structure of kernel should be pageable
- [x] e. The page fault handler should not be pageable
- [x] f. The kernel's own page tables should not be pageable

The correct answers are: The kernel's own page tables should not be pageable, The page fault handler should not be pageable, The kernel must have some dedicated frames for it's own work, The disk driver and disk interrupt handler should not be pageable

Question **5**

Complete

Mark 0.25 out of 1.00

Order the following events, related to page fault handling, in correct order

1. Page fault handler detects that it's a page fault and not illegal memory access
2. Page fault interrupt is generated
3. Page fault handler in kernel starts executing
4. Other processes scheduled by scheduler
5. Disk read is issued
6. Page faulting process is made to wait in a queue
7. Empty frame is found
8. Disk interrupt handler runs
9. Disk Interrupt occurs
10. Page table of page faulted process is updated
11. Page faulted process is moved to ready-queue
12. MMU detects that a page table entry is marked "invalid"

The correct order for these items is as follows:

1. MMU detects that a page table entry is marked "invalid"
2. Page fault interrupt is generated
3. Page fault handler in kernel starts executing
4. Page fault handler detects that it's a page fault and not illegal memory access
5. Empty frame is found
6. Disk read is issued
7. Page faulting process is made to wait in a queue
8. Other processes scheduled by scheduler
9. Disk Interrupt occurs
10. Disk interrupt handler runs
11. Page table of page faulted process is updated
12. Page faulted process is moved to ready-queue

Question **6**

Complete

Mark 1.00 out of 1.00

Calculate the EAT in NANO-seconds (upto 2 decimal points) w.r.t. a page fault, given

Memory access time = 140 ns

Average page fault service time = 10 ms

Page fault rate = 0.9

Answer: 9000014

The correct answer is: 9000014.00

Question **7**

Complete

Mark 1.00 out of 1.00

For the reference string
3 4 3 5 2

the number of page faults (including initial ones) using

FIFO replacement and 2 page frames is :

4

FIFO replacement and 3 page frames is :

4

Question **8**

Complete

Mark 1.00 out of 1.00

W.r.t the figure given below, mark the given statements as True or False.

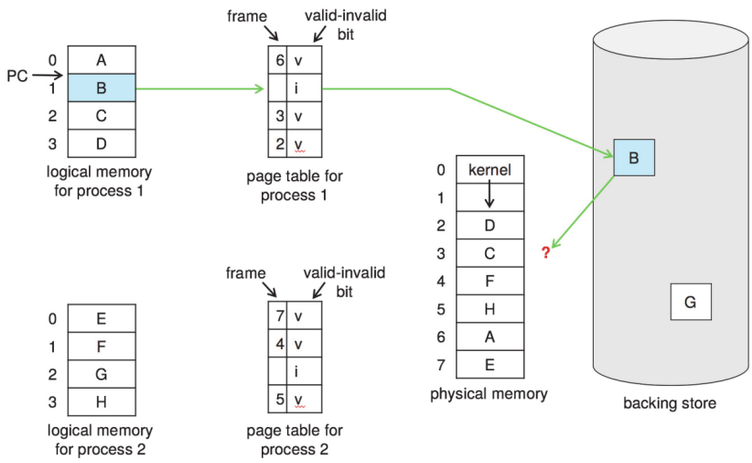


Figure 10.9 Need for page replacement.

True	False	
◉	○	Handling this scenario demands two disk I/Os
◉	○	Kernel occupies two page frames
◉	○	Local replacement means chose any of the frames 2, 3, 6
○	◉	Local replacement means chose any of the frame from 2 to 7
◉	○	Global replacement means chose any of the frame from 2 to 7
○	◉	Global replacement means chose any of the frame from 0 to 7
◉	○	Page 1 of process 1 needs a replacement
◉	○	The kernel's pages can not used for replacement if kernel is not pageable.

Handling this scenario demands two disk I/Os: True
Kernel occupies two page frames: True
Local replacement means chose any of the frames 2, 3, 6: True
Local replacement means chose any of the frame from 2 to 7: False
Global replacement means chose any of the frame from 2 to 7: True
Global replacement means chose any of the frame from 0 to 7: False
Page 1 of process 1 needs a replacement: True
The kernel's pages can not used for replacement if kernel is not pageable.: True

Question **9**

Complete

Mark 0.60 out of 1.00

Given below is the "maps" file for a particular instance of "vim.basic" process.

Mark the given statements as True or False, w.r.t. the contents of the map file.

```
55a43501b000-55a435049000 r--p 00000000 103:05 917529                    /usr/bin/vim.basic
55a435049000-55a435248000 r-xp 0002e000 103:05 917529                    /usr/bin/vim.basic
55a435248000-55a4352b6000 r--p 0022d000 103:05 917529                    /usr/bin/vim.basic
55a4352b7000-55a4352c5000 r--p 0029b000 103:05 917529                    /usr/bin/vim.basic
55a4352c5000-55a4352e2000 rw-p 002a9000 103:05 917529                    /usr/bin/vim.basic
55a4352e2000-55a4352f0000 rw-p 00000000 00:00 0
55a436bc9000-55a436e5b000 rw-p 00000000 00:00 0                          [heap]
7f275b0a3000-7f275b0a6000 r--p 00000000 103:05 917901                    /usr/lib/x86_64-linux-
gnu/libnss_files-2.31.so
7f275b0a6000-7f275b0ad000 r-xp 00003000 103:05 917901                    /usr/lib/x86_64-linux-
gnu/libnss_files-2.31.so
7f275b0ad000-7f275b0af000 r--p 0000a000 103:05 917901                    /usr/lib/x86_64-linux-
gnu/libnss_files-2.31.so
7f275b0af000-7f275b0b0000 r--p 0000b000 103:05 917901                    /usr/lib/x86_64-linux-
gnu/libnss_files-2.31.so
7f275b0b0000-7f275b0b1000 rw-p 0000c000 103:05 917901                    /usr/lib/x86_64-linux-
gnu/libnss_files-2.31.so
7f275b0b1000-7f275b0b7000 rw-p 00000000 00:00 0
7f275b0b7000-7f275b8f5000 r--p 00000000 103:05 925247                    /usr/lib/locale/locale-archive
7f275b8f5000-7f275b8fa000 rw-p 00000000 00:00 0
7f275b8fa000-7f275b8fc000 r--p 00000000 103:05 924216                    /usr/lib/x86_64-linux-
gnu/libogg.so.0.8.4
7f275b8fc000-7f275b901000 r-xp 00002000 103:05 924216                    /usr/lib/x86_64-linux-
gnu/libogg.so.0.8.4
7f275b901000-7f275b904000 r--p 00007000 103:05 924216                    /usr/lib/x86_64-linux-
gnu/libogg.so.0.8.4
7f275b904000-7f275b905000 ---p 0000a000 103:05 924216                    /usr/lib/x86_64-linux-
gnu/libogg.so.0.8.4
7f275b905000-7f275b906000 r--p 0000a000 103:05 924216                    /usr/lib/x86_64-linux-
gnu/libogg.so.0.8.4
7f275b906000-7f275b907000 rw-p 0000b000 103:05 924216                    /usr/lib/x86_64-linux-
gnu/libogg.so.0.8.4
7f275b907000-7f275b90a000 r--p 00000000 103:05 924627                    /usr/lib/x86_64-linux-
gnu/libvorbis.so.0.4.8
7f275b90a000-7f275b921000 r-xp 00003000 103:05 924627                    /usr/lib/x86_64-linux-
gnu/libvorbis.so.0.4.8
7f275b921000-7f275b932000 r--p 0001a000 103:05 924627                    /usr/lib/x86_64-linux-
gnu/libvorbis.so.0.4.8
7f275b932000-7f275b933000 ---p 0002b000 103:05 924627                    /usr/lib/x86_64-linux-
gnu/libvorbis.so.0.4.8
7f275b933000-7f275b934000 r--p 0002b000 103:05 924627                    /usr/lib/x86_64-linux-
gnu/libvorbis.so.0.4.8
7f275b934000-7f275b935000 rw-p 0002c000 103:05 924627                    /usr/lib/x86_64-linux-
gnu/libvorbis.so.0.4.8
7f275b935000-7f275b937000 rw-p 00000000 00:00 0
7f275b937000-7f275b938000 r--p 00000000 103:05 917914                    /usr/lib/x86_64-linux-
gnu/libutil-2.31.so
7f275b938000-7f275b939000 r-xp 00001000 103:05 917914                    /usr/lib/x86_64-linux-
gnu/libutil-2.31.so
7f275b939000-7f275b93a000 r--p 00002000 103:05 917914                    /usr/lib/x86_64-linux-
gnu/libutil-2.31.so
7f275b93a000-7f275b93b000 r--p 00002000 103:05 917914                    /usr/lib/x86_64-linux-
gnu/libutil-2.31.so
7f275b93b000-7f275b93c000 rw-p 00003000 103:05 917914                    /usr/lib/x86_64-linux-
gnu/libutil-2.31.so
7f275b93c000-7f275b93e000 r--p 00000000 103:05 915906                    /usr/lib/x86_64-linux-
gnu/libz.so.1.2.11
7f275b93e000-7f275b94f000 r-xp 00002000 103:05 915906                    /usr/lib/x86_64-linux-
gnu/libz.so.1.2.11
7f275b94f000-7f275b955000 r--p 00013000 103:05 915906                    /usr/lib/x86_64-linux-
gnu/libz.so.1.2.11
7f275b955000-7f275b956000 ---p 00019000 103:05 915906                    /usr/lib/x86_64-linux-
gnu/libz.so.1.2.11
7f275b956000-7f275b957000 r--p 00019000 103:05 915906                    /usr/lib/x86_64-linux-
gnu/libz.so.1.2.11
```

```
7f275b957000-7f275b958000 rw-p 0001a000 103:05 915906                    /usr/lib/x86_64-linux-
gnu/libz.so.1.2.11
7f275b958000-7f275b95c000 r--p 00000000 103:05 923645                    /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11
7f275b95c000-7f275b978000 r-xp 00004000 103:05 923645                    /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11
7f275b978000-7f275b982000 r--p 00020000 103:05 923645                    /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11
7f275b982000-7f275b983000 ---p 0002a000 103:05 923645                    /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11
7f275b983000-7f275b985000 r--p 0002a000 103:05 923645                    /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11
7f275b985000-7f275b986000 rw-p 0002c000 103:05 923645                    /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11
7f275b986000-7f275b988000 r--p 00000000 103:05 924057                    /usr/lib/x86_64-linux-
gnu/libltdl.so.7.3.1
7f275b988000-7f275b98d000 r-xp 00002000 103:05 924057                    /usr/lib/x86_64-linux-
gnu/libltdl.so.7.3.1
7f275b98d000-7f275b98f000 r--p 00007000 103:05 924057                    /usr/lib/x86_64-linux-
gnu/libltdl.so.7.3.1
7f275b98f000-7f275b990000 r--p 00008000 103:05 924057                    /usr/lib/x86_64-linux-
gnu/libltdl.so.7.3.1
7f275b990000-7f275b991000 rw-p 00009000 103:05 924057                    /usr/lib/x86_64-linux-
gnu/libltdl.so.7.3.1
7f275b991000-7f275b995000 r--p 00000000 103:05 921934                    /usr/lib/x86_64-linux-
gnu/libtdb.so.1.4.3
7f275b995000-7f275b9a3000 r-xp 00004000 103:05 921934                    /usr/lib/x86_64-linux-
gnu/libtdb.so.1.4.3
7f275b9a3000-7f275b9a9000 r--p 00012000 103:05 921934                    /usr/lib/x86_64-linux-
gnu/libtdb.so.1.4.3
7f275b9a9000-7f275b9aa000 r--p 00017000 103:05 921934                    /usr/lib/x86_64-linux-
gnu/libtdb.so.1.4.3
7f275b9aa000-7f275b9ab000 rw-p 00018000 103:05 921934                    /usr/lib/x86_64-linux-
gnu/libtdb.so.1.4.3
7f275b9ab000-7f275b9ad000 rw-p 00000000 00:00 0
7f275b9ad000-7f275b9af000 r--p 00000000 103:05 924631                    /usr/lib/x86_64-linux-
gnu/libvorbisfile.so.3.3.7
7f275b9af000-7f275b9b4000 r-xp 00002000 103:05 924631                    /usr/lib/x86_64-linux-
gnu/libvorbisfile.so.3.3.7
7f275b9b4000-7f275b9b5000 r--p 00007000 103:05 924631                    /usr/lib/x86_64-linux-
gnu/libvorbisfile.so.3.3.7
7f275b9b5000-7f275b9b6000 ---p 00008000 103:05 924631                    /usr/lib/x86_64-linux-
gnu/libvorbisfile.so.3.3.7
7f275b9b6000-7f275b9b7000 r--p 00008000 103:05 924631                    /usr/lib/x86_64-linux-
gnu/libvorbisfile.so.3.3.7
7f275b9b7000-7f275b9b8000 rw-p 00009000 103:05 924631                    /usr/lib/x86_64-linux-
gnu/libvorbisfile.so.3.3.7
7f275b9b8000-7f275b9ba000 r--p 00000000 103:05 924277                    /usr/lib/x86_64-linux-
gnu/libpcre2-8.so.0.9.0
7f275b9ba000-7f275ba1e000 r-xp 00002000 103:05 924277                    /usr/lib/x86_64-linux-
gnu/libpcre2-8.so.0.9.0
7f275ba1e000-7f275ba46000 r--p 00066000 103:05 924277                    /usr/lib/x86_64-linux-
gnu/libpcre2-8.so.0.9.0
7f275ba46000-7f275ba47000 r--p 0008d000 103:05 924277                    /usr/lib/x86_64-linux-
gnu/libpcre2-8.so.0.9.0
7f275ba47000-7f275ba48000 rw-p 0008e000 103:05 924277                    /usr/lib/x86_64-linux-
gnu/libpcre2-8.so.0.9.0
7f275ba48000-7f275ba6d000 r--p 00000000 103:05 917893                    /usr/lib/x86_64-linux-
gnu/libc-2.31.so
7f275ba6d000-7f275bbe5000 r-xp 00025000 103:05 917893                    /usr/lib/x86_64-linux-
gnu/libc-2.31.so
7f275bbe5000-7f275bc2f000 r--p 0019d000 103:05 917893                    /usr/lib/x86_64-linux-
gnu/libc-2.31.so
7f275bc2f000-7f275bc30000 ---p 001e7000 103:05 917893                    /usr/lib/x86_64-linux-
gnu/libc-2.31.so
7f275bc30000-7f275bc33000 r--p 001e7000 103:05 917893                    /usr/lib/x86_64-linux-
gnu/libc-2.31.so
7f275bc33000-7f275bc36000 rw-p 001ea000 103:05 917893                    /usr/lib/x86_64-linux-
gnu/libc-2.31.so
7f275bc36000-7f275bc3a000 rw-p 00000000 00:00 0
7f275bc3a000-7f275bc41000 r--p 00000000 103:05 917906                    /usr/lib/x86_64-linux-
gnu/libpthread-2.31.so
7f275bc41000-7f275bc52000 r-xp 00007000 103:05 917906                    /usr/lib/x86_64-linux-
```

```
gnu/libpthread-2.31.so
7f275bc52000-7f275bc57000 r--p 00018000 103:05 917906                    /usr/lib/x86_64-linux-
gnu/libpthread-2.31.so
7f275bc57000-7f275bc58000 r--p 0001c000 103:05 917906                    /usr/lib/x86_64-linux-
gnu/libpthread-2.31.so
7f275bc58000-7f275bc59000 rw-p 0001d000 103:05 917906                    /usr/lib/x86_64-linux-
gnu/libpthread-2.31.so
7f275bc59000-7f275bc5d000 rw-p 00000000 00:00 0
7f275bc5d000-7f275bcce000 r--p 00000000 103:05 917016                    /usr/lib/x86_64-linux-
gnu/libpython3.8.so.1.0
7f275bcce000-7f275bf29000 r-xp 00071000 103:05 917016                    /usr/lib/x86_64-linux-
gnu/libpython3.8.so.1.0
7f275bf29000-7f275c142000 r--p 002cc000 103:05 917016                    /usr/lib/x86_64-linux-
gnu/libpython3.8.so.1.0
7f275c142000-7f275c143000 ---p 004e5000 103:05 917016                    /usr/lib/x86_64-linux-
gnu/libpython3.8.so.1.0
7f275c143000-7f275c149000 r--p 004e5000 103:05 917016                    /usr/lib/x86_64-linux-
gnu/libpython3.8.so.1.0
7f275c149000-7f275c190000 rw-p 004eb000 103:05 917016                    /usr/lib/x86_64-linux-
gnu/libpython3.8.so.1.0
7f275c190000-7f275c1b3000 rw-p 00000000 00:00 0
7f275c1b3000-7f275c1b4000 r--p 00000000 103:05 917894                    /usr/lib/x86_64-linux-
gnu/libdl-2.31.so
7f275c1b4000-7f275c1b6000 r-xp 00001000 103:05 917894                    /usr/lib/x86_64-linux-
gnu/libdl-2.31.so
7f275c1b6000-7f275c1b7000 r--p 00003000 103:05 917894                    /usr/lib/x86_64-linux-
gnu/libdl-2.31.so
7f275c1b7000-7f275c1b8000 r--p 00003000 103:05 917894                    /usr/lib/x86_64-linux-
gnu/libdl-2.31.so
7f275c1b8000-7f275c1b9000 rw-p 00004000 103:05 917894                    /usr/lib/x86_64-linux-
gnu/libdl-2.31.so
7f275c1b9000-7f275c1bb000 rw-p 00000000 00:00 0
7f275c1bb000-7f275c1c0000 r-xp 00000000 103:05 923815                    /usr/lib/x86_64-linux-
gnu/libgpm.so.2
7f275c1c0000-7f275c3bf000 ---p 00005000 103:05 923815                    /usr/lib/x86_64-linux-
gnu/libgpm.so.2
7f275c3bf000-7f275c3c0000 r--p 00004000 103:05 923815                    /usr/lib/x86_64-linux-
gnu/libgpm.so.2
7f275c3c0000-7f275c3c1000 rw-p 00005000 103:05 923815                    /usr/lib/x86_64-linux-
gnu/libgpm.so.2
7f275c3c1000-7f275c3c3000 r--p 00000000 103:05 923315                    /usr/lib/x86_64-linux-
gnu/libacl.so.1.1.2253
7f275c3c3000-7f275c3c8000 r-xp 00002000 103:05 923315                    /usr/lib/x86_64-linux-
gnu/libacl.so.1.1.2253
7f275c3c8000-7f275c3ca000 r--p 00007000 103:05 923315                    /usr/lib/x86_64-linux-
gnu/libacl.so.1.1.2253
7f275c3ca000-7f275c3cb000 r--p 00008000 103:05 923315                    /usr/lib/x86_64-linux-
gnu/libacl.so.1.1.2253
7f275c3cb000-7f275c3cc000 rw-p 00009000 103:05 923315                    /usr/lib/x86_64-linux-
gnu/libacl.so.1.1.2253
7f275c3cc000-7f275c3cf000 r--p 00000000 103:05 923446                    /usr/lib/x86_64-linux-
gnu/libcanberra.so.0.2.5
7f275c3cf000-7f275c3d9000 r-xp 00003000 103:05 923446                    /usr/lib/x86_64-linux-
gnu/libcanberra.so.0.2.5
7f275c3d9000-7f275c3dd000 r--p 0000d000 103:05 923446                    /usr/lib/x86_64-linux-
gnu/libcanberra.so.0.2.5
7f275c3dd000-7f275c3de000 r--p 00010000 103:05 923446                    /usr/lib/x86_64-linux-
gnu/libcanberra.so.0.2.5
7f275c3de000-7f275c3df000 rw-p 00011000 103:05 923446                    /usr/lib/x86_64-linux-
gnu/libcanberra.so.0.2.5
7f275c3df000-7f275c3e5000 r--p 00000000 103:05 924431                    /usr/lib/x86_64-linux-
gnu/libselinux.so.1
7f275c3e5000-7f275c3fe000 r-xp 00006000 103:05 924431                    /usr/lib/x86_64-linux-
gnu/libselinux.so.1
7f275c3fe000-7f275c405000 r--p 0001f000 103:05 924431                    /usr/lib/x86_64-linux-
gnu/libselinux.so.1
7f275c405000-7f275c406000 ---p 00026000 103:05 924431                    /usr/lib/x86_64-linux-
gnu/libselinux.so.1
7f275c406000-7f275c407000 r--p 00026000 103:05 924431                    /usr/lib/x86_64-linux-
gnu/libselinux.so.1
7f275c407000-7f275c408000 rw-p 00027000 103:05 924431                    /usr/lib/x86_64-linux-
gnu/libselinux.so.1
7f275c408000-7f275c40a000 rw-p 00000000 00:00 0
```

```
7f275c40a000-7f275c418000 r--p 00000000 103:05 924540                    /usr/lib/x86_64-linux-
gnu/libtinfo.so.6.2
7f275c418000-7f275c427000 r-xp 0000e000 103:05 924540                    /usr/lib/x86_64-linux-
gnu/libtinfo.so.6.2
7f275c427000-7f275c435000 r--p 0001d000 103:05 924540                    /usr/lib/x86_64-linux-
gnu/libtinfo.so.6.2
7f275c435000-7f275c439000 r--p 0002a000 103:05 924540                    /usr/lib/x86_64-linux-
gnu/libtinfo.so.6.2
7f275c439000-7f275c43a000 rw-p 0002e000 103:05 924540                    /usr/lib/x86_64-linux-
gnu/libtinfo.so.6.2
7f275c43a000-7f275c449000 r--p 00000000 103:05 917895                    /usr/lib/x86_64-linux-
gnu/libm-2.31.so
7f275c449000-7f275c4f0000 r-xp 0000f000 103:05 917895                    /usr/lib/x86_64-linux-
gnu/libm-2.31.so
7f275c4f0000-7f275c587000 r--p 000b6000 103:05 917895                    /usr/lib/x86_64-linux-
gnu/libm-2.31.so
7f275c587000-7f275c588000 r--p 0014c000 103:05 917895                    /usr/lib/x86_64-linux-
gnu/libm-2.31.so
7f275c588000-7f275c589000 rw-p 0014d000 103:05 917895                    /usr/lib/x86_64-linux-
gnu/libm-2.31.so
7f275c589000-7f275c58b000 rw-p 00000000 00:00 0
7f275c5ae000-7f275c5af000 r--p 00000000 103:05 917889                    /usr/lib/x86_64-linux-gnu/ld-
2.31.so
7f275c5af000-7f275c5d2000 r-xp 00001000 103:05 917889                    /usr/lib/x86_64-linux-gnu/ld-
2.31.so
7f275c5d2000-7f275c5da000 r--p 00024000 103:05 917889                    /usr/lib/x86_64-linux-gnu/ld-
2.31.so
7f275c5db000-7f275c5dc000 r--p 0002c000 103:05 917889                    /usr/lib/x86_64-linux-gnu/ld-
2.31.so
7f275c5dc000-7f275c5dd000 rw-p 0002d000 103:05 917889                    /usr/lib/x86_64-linux-gnu/ld-
2.31.so
7f275c5dd000-7f275c5de000 rw-p 00000000 00:00 0
7ffd22d2f000-7ffd22d50000 rw-p 00000000 00:00 0                          [stack]
7ffd22db0000-7ffd22db4000 r--p 00000000 00:00 0                          [vvar]
7ffd22db4000-7ffd22db6000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
```

True	False	
◉	○	The size of the stack is one page
◉	○	This is a virtual memory map (not physical memory map)
◉	○	The size of the heap is one page
◉	○	vim.basic uses the math library
◉	○	The 5th entry 55a4352c5000-55a4352e2000 **may** correspond to "data" of the vim.basic

The size of the stack is one page: False
This is a virtual memory map (not physical memory map): True
The size of the heap is one page: False
vim.basic uses the math library: True
The 5th entry 55a4352c5000-55a4352e2000 **may** correspond to "data" of the vim.basic: True

Question **10**

Complete

Mark 0.50 out of 0.50

Map the parts of a C code to the memory regions they are related to

global initialized variables	data
malloced memory	heap
static variables	data
functions	code
global un-initialized variables	bss
function arguments	stack
local variables	stack

The correct answer is: global initialized variables → data, malloced memory → heap, static variables → data, functions → code, global un-initialized variables → bss, function arguments → stack, local variables → stack

Question **11**

Complete

Mark 1.00 out of 1.00

Page sizes are a power of 2 because

Select one:

- a. Power of 2 calculations are highly efficient
- b. operating system calculations happen using power of 2
- c. Certain bits are reserved for offset in logical address. Hence page size = 2^(32 - no.of offset bits)
- d. MMU only understands numbers that are power of 2
- e. Certain bits are reserved for offset in logical address. Hence page size = 2^(no.of offset bits)

The correct answer is: Certain bits are reserved for offset in logical address. Hence page size = 2^(no.of offset bits)

Question **12**

Complete

Mark 0.50 out of 0.50

Map the technique with it's feature/problem

dynamic linking	small executable file
static loading	wastage of physical memory
dynamic loading	allocate memory only if needed
static linking	large executable file

The correct answer is: dynamic linking → small executable file, static loading → wastage of physical memory, dynamic loading → allocate memory only if needed, static linking → large executable file

Question **13**

Complete

Mark 1.00 out of 1.00

Assuming a 8- KB page size, what is the page numbers for the address 803160 reference in decimal :

(give answer also in decimal)

Answer: 98

The correct answer is: 98

Question **14**

Complete

Mark 0.43 out of 1.00

Suppose two processes share a library between them. The library consists of 5 pages, and these 5 pages are mapped to frames 9, 15, 23, 4, 7 respectively. Process P1 has got 6 pages, first 3 of which consist of process's own code/data and 3 correspond to library's pages 0, 2, 4. Process P2 has got 7 pages, first 3 of which consist of processe's own code/data and remaining 4 correspond to library's pages 0, 1, 3, 4. Fill in the blanks for page table entries of P1 and P2.

Page table of P2, Page 3: 9

Page table of P2, Page 4: 15

Page table of P2, Page 0: 5

Page table of P1, Page 3: 9

Page table of P1, Page 5: 7

Page table of P1, Page 4: 23

Page table of P2, Page 1: 6

The correct answer is: Page table of P2, Page 3 → 4, Page table of P2, Page 4 → 7, Page table of P2, Page 0 → 9, Page table of P1, Page 3 → 9, Page table of P1, Page 5 → 7, Page table of P1, Page 4 → 23, Page table of P2, Page 1 → 15

Question **15**

Complete

Mark 1.00 out of 1.00

Given six memory partitions of 300 KB , 600 KB , 350 KB , 200 KB , 750 KB , and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB and 500 KB (in order)?

best fit 115 KB: 125 KB

worst fit 500 KB: 635 KB

first fit 500 KB: 600 KB

best fit 500 KB: 600 KB

first fit 115 KB: 300 KB

worst fit 115 KB: 750 KB

The correct answer is: best fit 115 KB → 125 KB, worst fit 500 KB → 635 KB, first fit 500 KB → 600 KB, best fit 500 KB → 600 KB, first fit 115 KB → 300 KB, worst fit 115 KB → 750 KB

Question **16**

Complete

Mark 0.25 out of 1.00

Select all the correct statements, w.r.t. Copy on Write

- [ ] a. Vfork() assumes that there will be no write, but rather exec()
- [x] b. Fork() used COW technique to improve performance of new process creation.
- [x] c. COW helps us save memory
- [ ] d. use of COW during fork() is useless if child called exit()
- [x] e. If either parent or child modifies a COW-page, then a copy of the page is made and page table entry is updated
- [x] f. use of COW during fork() is useless if exec() is called by the child

The correct answers are: Fork() used COW technique to improve performance of new process creation., If either parent or child modifies a COW-page, then a copy of the page is made and page table entry is updated, COW helps us save memory, Vfork() assumes that there will be no write, but rather exec()

◄ (Code) mmap related programs

Jump to...

Points from Mid-term feedback ►

Started on	Monday, 7 March 2022, 7:02:48 PM
State	Finished
Completed on	Monday, 7 March 2022, 8:20:00 PM
Time taken	1 hour 17 mins
Grade	**11.67** out of 15.00 (**78**%)

Question **1**

Complete

Mark 1.00 out of 1.00

Map the virtual address to physical address in xv6

KERNLINK	0x100000
KERNBASE	0
80108000	0x108000
0xFE000000	0xFE000000

The correct answer is: KERNLINK → 0x100000, KERNBASE → 0, 80108000 → 0x108000, 0xFE000000 → 0xFE000000

Question **2**

Complete

Mark 1.00 out of 1.00

Why is there a call to kinit2? Why is it not merged with knit1?

- a. knit2 refers to virtual addresses beyond 4MB, which are not mapped before kvalloc() is called
- b. call to seginit() makes it possible to actually use PHYSTOP in argument to kinit2()
- c. Because there is a limit on the values that the argumets to knit1() can take.
- d. When kinit1() is called there is a need for few page frames, but later knit2() is called to serve need of more page frames

The correct answer is: knit2 refers to virtual addresses beyond 4MB, which are not mapped before kvalloc() is called

Question **3**

Complete

Mark 1.00 out of 1.00

Which of the following is done by mappages()?

- a. allocate page directory if required
- b. allocate page table if required
- c. allocate page frame if required
- d. create page table mappings for the range given by "va" and "va + size"
- e. create page table mappings to the range given by "pa" and "pa + size"

The correct answers are: create page table mappings for the range given by "va" and "va + size", allocate page table if required, create page table mappings to the range given by "pa" and "pa + size"

Question **4**

Complete

Mark 0.00 out of 1.00

Select all the correct statements about initcode

- [x] a. code of initcode is loaded in memory by the kernel during userinit()
- [ ] b. code of initcode is loaded at virtual address 0
- [ ] c. initcode essentially calls exec("/init",...)
- [x] d. the size of 'initcode' is 2c
- [x] e. code of 'initcode' is loaded along with the kernel during booting
- [x] f. initcode is the 'init' process
- [ ] g. The data and stack of initcode is mapped to one single page in userinit()

The correct answers are: code of 'initcode' is loaded along with the kernel during booting, the size of 'initcode' is 2c, The data and stack of initcode is mapped to one single page in userinit(), initcode essentially calls exec("/init",...)

Question **5**

Complete

Mark 1.50 out of 1.50

Which of the following is DONE by allocproc() ?

- [x] a. allocate kernel stack for the process
- [x] b. allocate PID to the process
- [x] c. Select an UNUSED struct proc for use
- [ ] d. setup kernel memory mappings for the process
- [x] e. ensure that the process starts in forkret()
- [x] f. setup the trapframe and context pointers appropriately
- [ ] g. setup the contents of the trapframe of the process properly
- [ ] h. ensure that the process starts in trapret()

The correct answers are: Select an UNUSED struct proc for use, allocate PID to the process, allocate kernel stack for the process, setup the trapframe and context pointers appropriately, ensure that the process starts in forkret()

Question **6**

Complete

Mark 0.67 out of 2.00

exec() does this: curproc->tf->eip = elf.entry, but userinit() does this: p->tf->eip = 0; Select all the statements from below, that collectively explain this

- [ ] a. the code of 'initcode' is loaded at physical address 0
- [x] b. exec() loads from ELF file and the address of first instruction to be executed is given by 'entry'
- [ ] c. elf.entry is anyways 0, so both statements mean the same
- [ ] d. the initcode is created using objcopy, which discards all relocation information and symbols (like entry)
- [x] e. the 'entry' in initcode is anyways 0
- [x] f. In userinit() the function inituvm() has mapped the code of 'initcode' to be starting at virtual address 0

The correct answers are: exec() loads from ELF file and the address of first instruction to be executed is given by 'entry', In userinit() the function inituvm() has mapped the code of 'initcode' to be starting at virtual address 0, the initcode is created using objcopy, which discards all relocation information and symbols (like entry)

Question **7**

Complete

Mark 1.00 out of 1.00

What does userinit() do ?

- a. sets up the 'initcode' process to start execution in forkret ()
- b. sets up the 'init' process to start execution in forkret()
- c. initializes the process 'init' and starts executing it
- d. sets up the 'initcode' process to start execution in forkret()
- e. initializes the users
- f. sets up the 'initcode' process to start execution in trapret()

The correct answer is: sets up the 'initcode' process to start execution in forkret()

Question **8**

Complete

Mark 0.00 out of 1.00

The approximate number of page frames created by kinit1 is

- a. 2000
- b. 3000
- c. 4
- d. 1000
- e. 16
- f. 10
- g. 4000

The correct answer is: 3000

Question **9**

Complete

Mark 1.00 out of 1.00

Does exec() code around clearptau() lead to wastage of one page frame?

- a. yes
- b. no

The correct answer is: yes

Question **10**

Complete

Mark 1.00 out of 1.00

Select the statement that most correctly describes what setupkvm() does

- a. creates a 2-level page table for the use of the kernel, as specified in gdtdesc
- b. creates a 1-level page table for the use by the kernel, as specified in kmap[] global array
- c. creates a 2-level page table setup with virtual->physical mappings specified in the kmap[] global arrray
- d. creates a 2-level page table setup with virtual->physical mappings specified in the kmap[] global arrray and makes kpgdir point to it

The correct answer is: creates a 2-level page table setup with virtual->physical mappings specified in the kmap[] global arrray

Question **11**

Complete

Mark 1.50 out of 1.50

Arrange the following in the correct order of execution (w.r.t. 'init')

initcode() returns from trapret()	7
userinit() is called	1
'initcode' process is marked RUNNABLE	3
mpmain() calls scheduler()	4
'initcode' struct proc is created	2
initcode() calls exec("/init", ...)	8
initcode() returns in forkret()	6
scheduler() schedules initcode() process	5

The correct answer is: initcode() returns from trapret() → 7, userinit() is called → 1, 'initcode' process is marked RUNNABLE → 3, mpmain() calls scheduler() → 4, 'initcode' struct proc is created → 2, initcode() calls exec("/init", ...) → 8, initcode() returns in forkret() → 6, scheduler() schedules initcode() process → 5

Question **12**

Complete

Mark 1.00 out of 1.00

What does seginit() do?

- a. Nothing significant, just repetition of earlier GDT setup but with kernel page table allocated now
- b. Nothing significant, just repetition of earlier GDT setup but with free frames list created now
- c. Adds two additional entries to GDT corresponding to Code and Data segments, but to be used in privilege level 3
- d. Nothing significant, just repetition of earlier GDT setup but with 2-level paging setup done
- e. Adds two additional entries to GDT corresponding to Code and Data segments, but to be used in privilege level 0

The correct answer is: Adds two additional entries to GDT corresponding to Code and Data segments, but to be used in privilege level 3

Question **13**

Complete

Mark 1.00 out of 1.00

The variable 'end' used as argument to kinit1 has the value

- a. 8010a48c
- b. 80102da0
- c. 81000000
- d. 801154a8
- e. 80110000
- f. 80000000

The correct answer is: 801154a8

◄ Questions for test on kalloc/kfree/kvmalloc, etc.

Jump to...

(Optional Assignment) Slab allocator in xv6 ►

Started on	Monday, 24 January 2022, 7:11:55 PM
State	Finished
Completed on	Monday, 24 January 2022, 9:11:03 PM
Time taken	1 hour 59 mins
Grade	**13.10** out of 20.00 (**66**%)

Question **1**

Complete

Mark 1.00 out of 1.00

Rank the following storage systems from slowest (first) to fastest(last)

You can drag and drop the items below/above each other.

- Magnetic tapes
- Optical disk
- Hard-disk drives
- Nonvolatile memory
- Main memory
- Cache
- Registers

Question **2**

Complete

Mark 2.00 out of 2.00

Match the program with it's output (ignore newlines in the output. Just focus on the count of the number of 'hi')

main() { int i = fork(); if(i == 0) execl("/usr/bin/echo", "/usr/bin/echo", "hi\n", NULL); }	hi
main() { fork(); execl("/usr/bin/echo", "/usr/bin/echo", "hi\n", NULL); }	hi hi
main() { execl("/usr/bin/echo", "/usr/bin/echo", "hi\n", NULL); }	hi
main() { int i = NULL; fork(); printf("hi\n"); }	hi hi

The correct answer is: main() { int i = fork(); if(i == 0) execl("/usr/bin/echo", "/usr/bin/echo", "hi\n", NULL); } → hi, main() { fork(); execl("/usr/bin/echo", "/usr/bin/echo", "hi\n", NULL); } → hi hi, main() { execl("/usr/bin/echo", "/usr/bin/echo", "hi\n", NULL); } → hi, main() { int i = NULL; fork(); printf("hi\n"); } → hi hi

Question **3**

Complete

Mark 1.00 out of 1.00

How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) ?

Select one:

- a. It prohibits invocation of kernel code completely, if a user program is running
- b. It prohibits one process from accessing other process's memory
- c. It disallows hardware interrupts when a process is running
- d. It prohibits a user mode process from running privileged instructions

The correct answer is: It prohibits a user mode process from running privileged instructions

Question **4**

Complete

Mark 1.00 out of 1.00

Why should a program exist in memory before it starts executing ?

- a. Because the variables of the program are stored in memory
- b. Because the memory is volatile
- c. Because the hard disk is a slow medium
- d. Becase the processor can run instructions and access data only from memory

The correct answer is: Becase the processor can run instructions and access data only from memory

Question **5**

Complete

Mark 0.50 out of 0.50

Order the following events in boot process (from 1 onwards)

BIOS	1
OS	3
Boot loader	2
Init	4
Login interface	5
Shell	6

The correct answer is: BIOS → 1, OS → 3, Boot loader → 2, Init → 4, Login interface → 5, Shell → 6

Question **6**

Complete

Mark 0.50 out of 0.50

Compare multiprogramming with multitasking

- a. A multitasking system is not necessarily multiprogramming
- b. A multiprogramming system is not necessarily multitasking

The correct answer is: A multiprogramming system is not necessarily multitasking

Question **7**

Complete

Mark 1.50 out of 2.00

Which of the following are NOT a part of job of a typical compiler?

- a. Suggest alternative pieces of code that can be written
- b. Invoke the linker to link the function calls with their code, extern globals with their declaration
- c. Process the # directives in a C program
- d. Convert high level langauge code to machine code
- e. Check the program for logical errors
- f. Check the program for syntactical errors

The correct answers are: Check the program for logical errors, Suggest alternative pieces of code that can be written

Question **8**

Complete

Mark 0.50 out of 0.50

Is the terminal a part of the kernel on GNU/Linux systems?

- a. no
- b. yes

The correct answer is: no

Question **9**

Complete

Mark 0.00 out of 2.00

Select all the correct statements about calling convention on x86 32-bit.

- [x] a. The two lines in the beginning of each function, "push %ebp; mov %esp, %ebp", create space for local variables
- [ ] b. Return address is one location above the ebp
- [ ] c. Parameters may be passed in registers or on stack
- [x] d. The return value is either stored on the stack or returned in the eax register
- [x] e. Space for local variables is allocated by substracting the stack pointer inside the code of the caller function
- [x] f. Space for local variables is allocated by substracting the stack pointer inside the code of the called function
- [ ] g. during execution of a function, ebp is pointing to the old ebp
- [x] h. The ebp pointers saved on the stack constitute a chain of activation records
- [x] i. Parameters may be passed in registers or on stack
- [x] j. Compiler may allocate more memory on stack than needed
- [ ] k. Paramters are pushed on the stack in left-right order

The correct answers are: Compiler may allocate more memory on stack than needed, Parameters may be passed in registers or on stack, Parameters may be passed in registers or on stack, Return address is one location above the ebp, during execution of a function, ebp is pointing to the old ebp, Space for local variables is allocated by substracting the stack pointer inside the code of the called function, The ebp pointers saved on the stack constitute a chain of activation records

Question **10**

Complete

Mark 1.00 out of 1.00

```
int value = 5;
int main()
{
   pid_t pid;
   pid = fork();
   if (pid == 0) { /* child process */
      value += 15;
      return 0;
   }
   else if (pid > 0) { /* parent process */
      wait(NULL);
      printf("%d", value); /* LINE A */
   }
   return 0;
}
```

What's the value printed here at LINE A?

Answer: 5

The correct answer is: 5

Question **11**

Complete

Mark 0.00 out of 0.50

Is the command "cat README > done &" possible on xv6? (Note the & in the end)

- a. yes
- b. no

The correct answer is: yes

Question **12**

Complete

Mark 0.00 out of 2.00

xv6.img: bootblock kernel

```
    dd if=/dev/zero of=xv6.img count=10000
    dd if=bootblock of=xv6.img conv=notrunc
    dd if=kernel of=xv6.img seek=1 conv=notrunc
```

Consider above lines from the Makefile. Which of the following is incorrect?

- [x] a. The xv6.img is of the size 10,000 blocks of 512 bytes each and occupies upto 10,000 blocks on the disk.
- [x] b. xv6.img is the virtual processor used by the qemu emulator
- [x] c. The size of the xv6.img is nearly 5 MB
- [ ] d. The kernel is located at block-1 of the xv6.img
- [ ] e. The xv6.img is the virtual disk that is created by combining the bootblock and the kernel file.
- [ ] f. The size of xv6.img is exactly = (size of bootblock) + (size of kernel)
- [ ] g. The size of the kernel file is nearly 5 MB
- [x] h. The xv6.img is of the size 10,000 blocks of 512 bytes each and occupies 10,000 blocks on the disk.
- [x] i. The bootblock may be 512 bytes or less (looking at the Makefile instruction)
- [x] j. The bootblock is located on block-0 of the xv6.img
- [x] k. Blocks in xv6.img after kernel may be all zeroes.

The correct answers are: xv6.img is the virtual processor used by the qemu emulator, The xv6.img is of the size 10,000 blocks of 512 bytes each and occupies upto 10,000 blocks on the disk., The size of the kernel file is nearly 5 MB, The size of xv6.img is exactly = (size of bootblock) + (size of kernel)

Question **13**

Complete

Mark 1.00 out of 1.00

Match the register with the segment used with it.

ebp	ss
esi	ds
esp	ss
edi	es
eip	cs

The correct answer is: ebp → ss, esi → ds, esp → ss, edi → es, eip → cs

Question **14**

Complete

Mark 1.67 out of 2.00

Which of the following instructions should be privileged?

Select one or more:

- [ ] a. Access a general purpose register
- [x] b. Access I/O device.
- [x] c. Access memory management unit of the processor
- [x] d. Set value of timer.
- [ ] e. Set value of a memory location
- [ ] f. Read the clock.
- [x] g. Turn off interrupts.
- [ ] h. Switch from user to kernel mode.
- [x] i. Modify entries in device-status table

The correct answers are: Set value of timer., Access memory management unit of the processor, Turn off interrupts., Modify entries in device-status table, Access I/O device., Switch from user to kernel mode.

Question **15**

Complete

Mark 0.60 out of 1.00

Select all the correct statements about two modes of CPU operation

Select one or more:

- [x] a. Some instructions are allowed to run only in user mode, while all instructions can run in kernel mode
- [ ] b. The two modes are essential for a multitasking system
- [ ] c. There is an instruction like 'iret' to return from kernel mode to user mode
- [x] d. The two modes are essential for a multiprogramming system
- [x] e. The software interrupt instructions change the mode from user mode to kernel mode and jumps to predefined location simultaneously

The correct answers are: The two modes are essential for a multiprogramming system, The two modes are essential for a multitasking system, There is an instruction like 'iret' to return from kernel mode to user mode, The software interrupt instructions change the mode from user mode to kernel mode and jumps to predefined location simultaneously, Some instructions are allowed to run only in user mode, while all instructions can run in kernel mode

Question **16**

Complete

Mark 0.83 out of 2.00

Select all statements that correctly explain the use/purpose of system calls.

Select one or more:

- [ ] a. Run each instruction of an application program
- [x] b. Provide services for accessing files
- [ ] c. Handle ALL types of interrupts
- [ ] d. Allow I/O device access to user processes
- [x] e. Switch from user mode to kernel mode
- [x] f. Provide an environment for process creation
- [x] g. Handle exceptions like division by zero

The correct answers are: Switch from user mode to kernel mode, Provide services for accessing files, Allow I/O device access to user processes, Provide an environment for process creation

◄ uLearn Test Quiz (Fri 21 Jan)- Div1-T2

Jump to...

(Optional Assignment) Shell Programming(Conformance tests) ►

Started on	Wednesday, 9 February 2022, 7:00:45 PM
State	Finished
Completed on	Wednesday, 9 February 2022, 7:59:50 PM
Time taken	59 mins 5 secs
Grade	**8.00** out of 11.00 (**73**%)

Question **1**

Complete

Mark 1.00 out of 1.00

The right side of line of code "entry = (void(*)(void))(elf->entry)" means

- a. Get the "entry" in ELF structure and convert it into a void pointer
- b. Get the "entry" in ELF structure and convert it into a function pointer accepting no arguments and returning nothing
- c. Get the "entry" in ELF structure and convert it into a function void pointer
- d. Convert the "entry" in ELF structure into void

The correct answer is: Get the "entry" in ELF structure and convert it into a function pointer accepting no arguments and returning nothing

Question **2**

Complete

Mark 1.00 out of 1.00

The number of GDT entries setup during boot process of xv6 is

- a. 256
- b. 4
- c. 255
- d. 2
- e. 3
- f. 0

The correct answer is: 3

Question **3**

Not answered

Marked out of 0.50

code line, MMU setting: Match the line of xv6 code with the MMU setup employed

Answer:

The correct answer is: inb $0x64,%al

Question **4**

Complete

Mark 0.00 out of 1.00

The kernel ELF file contains how many Program headers?

- a. 9
- b. 4
- c. 10
- d. 3
- e. 2

The correct answer is: 3

Question **5**

Complete

Mark 0.00 out of 1.00

x86 provides which of the following type of memory management options?

- a. segmentation and one level paging
- b. segmentation only
- c. segmentation and one or two level paging
- d. segmentation or one or two level paging
- e. segmentation or paging
- f. segmentation and two level paging

The correct answer is: segmentation and one or two level paging

Question **6**

Complete

Mark 1.00 out of 1.00

ELF Magic number is

- a. 0xELFELFELF
- b. 0xFFFFFFFFF
- c. 0x464C457FL
- d. 0xELF
- e. 0x464C457FU
- f. 0
- g. 0x0x464CELF

The correct answer is: 0x464C457FU

Question **7**

Complete

Mark 1.00 out of 1.00

The ljmp instruction in general does

- a. change the CS and EIP to 32 bit mode, and jumps to next line of code
- b. change the CS and EIP to 32 bit mode
- c. change the CS and EIP to 32 bit mode, and jumps to new value of EIP
- d. change the CS and EIP to 32 bit mode, and jumps to kernel code

The correct answer is: change the CS and EIP to 32 bit mode, and jumps to new value of EIP

Question **8**

Complete

Mark 1.00 out of 1.00

The variable $stack in entry.S is

- a. located at the value given by %esp as setup by bootmain()
- b. a memory region allocated as a part of entry.S
- c. located at less than 0x7c00
- d. located at 0x7c00
- e. located at 0

The correct answer is: a memory region allocated as a part of entry.S

Question **9**

Complete

Mark 1.00 out of 1.00

Why is the code of entry() in Assembly and not in C?

- a. Because it needs to setup paging
- b. Because the kernel code must begin in assembly
- c. There is no particular reason, it could also be in C
- d. Because the symbol entry() is inside the ELF file

The correct answer is: Because it needs to setup paging

Question **10**

Not answered

Marked out of 0.50

Match the pairs of which action is taken by whom

Answer:

The correct answer is: kernel

Question **11**

Complete

Mark 1.00 out of 1.00

which of the following is not a difference between real mode and protected mode

- a. processor starts in real mode
- b. in real mode the segment is multiplied by 16, in protected mode segment is used as index in GDT
- c. in real mode the addressable memory is more than in protected mode
- d. in real mode the addressable memory is less than in protected mode
- e. in real mode general purpose registers are 16 bit, in protected mode they are 32 bit

The correct answer is: in real mode the addressable memory is more than in protected mode

Question **12**

Complete

Mark 1.00 out of 1.00

The kernel is loaded at Physical Address

- a. 0x80000000
- b. 0x00100000
- c. 0x80100000
- d. 0x0010000

The correct answer is: 0x00100000

◄ Homework questions: Basics of MM, xv6 booting

Jump to...

(Code) Files, redirection, dup, (IPC)pipe ►