Table of Contents

Shorter Questions ........................................................................................................................................ 1
Process State I............................................................................................................................................... 5
Process State II........................................................................................................................................... 6
Process State III......................................................................................................................................... 6
Process State IV.......................................................................................................................................... 6
Linux Three-Level Page Table I.............................................................................................................. 7
Linux Three-Level Page Table II........................................................................................................... 7
Linux Three-Level Page Table III........................................................................................................... 7
Buddy Algorithm I........................................................................................................................................ 8
Buddy Algorithm II...................................................................................................................................... 8
HardDisk (I/O)............................................................................................................................................ 9
Disk Partitions I.......................................................................................................................................... 9
Disk Partitions II....................................................................................................................................... 9
File System................................................................................................................................................ 10
Fork I.......................................................................................................................................................... 11
Fork II........................................................................................................................................................ 11
Fork III........................................................................................................................................................ 12
Fork IV........................................................................................................................................................ 12
MultiThreads.......................................................................................................................................... 13
Synchronization I...................................................................................................................................... 14
Synchronization II.................................................................................................................................... 15

Shorter Questions

a) Explain briefly, the **two basic functions** that Operating Systems perform!

b) One of the Operating Systems' basic function is to present the user with the equivalent of an extended machine. Explain what an **extended machine** is!

c) What is a **virtual machine**? Give an example/illustration!

d) One of the Operating Systems' basic function is managing resources. Explain what an **managing resources** is!

e) These following are fundamental principles of an Operating System:

   
   Explain briefly three (3) fundamental principles from the list above!

f) What is a **Real Time** system? Give an illustration!

g) What is a **Hard Real Time** system? Give an illustration!
h) What is a **Soft Real Time** system? Give an illustration!

i) What are the differences between a **System Program** and an **Application Program**?

j) Give an example of a **System Program**!

k) Give an example of a **Application Program**!

l) What are the differences between a **System Program** and a **System Call**? Give an illustration (eg. "creating a directory")!

m) How is Win32 API (Application Program Interface) related to a **System Call**.

n) Explain what a **Critical Region** is!

o) Explain what a **Race Condition** is!

p) Explain what a **Busy Waiting** is! How to overcome it?

q) What is a **Deadlock**? Explain briefly!

r) How does Unix handle the **Deadlock** problem? Explain briefly!

s) What is a **Starvation**? Explain briefly!

t) How does these following systems handle the **deadlock** problem:
   - Unix
   - Windows
   - JVM
   Explain briefly!

u) What is a binary semaphore?

v) Explain briefly, how to use binary semaphores for access control of a critical section!

w) What is a counting semaphore?

x) Explain briefly, how to use counting semaphores for access control of a resource with a finite number of instances?

y) Explain the differences between running a process in **kernel mode** and **user mode**?

z) Give two examples/illustration of running a process in "kernel mode".

aa) Give two examples/illustration of running a process in "user mode".

ab) Explain what "multi-programming" means. Give an example!

ac) Explain what "multi-users" means. Give an example!
ad) Explain what a "process table" is. Give an illustration!

ae) Explain what a "file system" is. Give an example!

af) Explain what a "pipe" is. Give an illustration!

ag) Explain what a "socket" is. Give an illustration!

ah) In a three state process model ("running", "blocked", and "ready"), explain briefly about each process state.

ai) In a three state process model ("running", "blocked", and "ready"), explain why a "running" state process transits to "blocked" state.

aj) In a three state process model ("running", "blocked", and "ready"), explain why a "running" state process transits to "ready" state.

ak) In a three state process model ("running", "blocked", and "ready"), explain why a "ready" state process transits to "running" state.

al) In a three state process model ("running", "blocked", and "ready"), explain why a "blocked" state process transits to "ready" state.

am) In a three state process model ("running", "blocked", and "ready"), explain why there is no "blocked" state process transits to "running" state.

an) In a three state process model ("running", "blocked", and "ready"), explain why there is no "ready" state process transits to "blocked" state.

ao) What is a "CPU bound" process? Give an illustration!

ap) What is a "I/O bound" process? Give an illustration!

aq) What is a "preemptive" process? Give an illustration!

ar) What is a "non-preemptive" process? Give an illustration!

as) Explain briefly the "Readers/Writers" problem. How to avoid "deadlock" in the "Readers/Writers" problem.

at) Explain briefly the "Readers/Writers" problem. Where is the "critical section" of the "Readers/Writers" problem.

au) Explain briefly the "Consumer/Producer" problem. How to avoid "deadlock" in the "Consumer/Producer" problem.

av) Explain briefly the Consumer/Producer problem. Where is the critical section of the Consumer/Producer problem.

aw) What are the differences and similarities between the Consumer/Producer problem and Readers/Writers problem? Explain briefly!
ax) Explain how a "preemptive" system can improve performance!

ay) What will improve, if more "RAM" is added to a system? Give illustrations!

az) What will improve, if the "CPU" of the system is replaced with a faster one? Give illustrations!

ba) What will improve, if the "DISK" of the system is replaced with a faster transfer rate? Give illustrations!

bb) What will improve, if the "I/O Bus" of the system is replaced with a faster transfer rate? Give illustrations!

bc) Which task should have more priority: writing to a disk or reading from a disk? Explain!

bd) Explain how a higher "RPM rate" can improve disk transfer rate!

be) Explain how a higher "disk density" can improve disk transfer rate!

bf) Explain how a DMA scheme can improve the system performance

bg) What is a "Hard Real Time System"? Give an example!

bh) What is a "Soft Real Time System"? Give an example!

bi) Compare the performance between a "pipe" and "file". Explain!

bj) Compare the performance between a "pipe" and "socket". Explain!

bk) Compare the performance between a "socket" and "file"? Explain!
Process State I

- At t=0, all processes (P₁, P₂, P₃, P₄) are in the "RDY" state.
- The "RUN/W" (Wait) state patterns of each process are as following:
  - P₁ (2, 9, 2, 9, 2, 9, ...)
  - P₂ (1, 9, 1, 9, 1, 9, ...)
  - P₃ (2, 6, 2, 6, 2, 6, ...)
  - P₄ (1, 6, 1, 6, 1, 6, ...)
- Only one process can be in the "RUN" state at any time.
- Many processes can be in the "W" and/or "RDY" states.
- The "RDY" to "RUN" transition rules are as following:
  - Priority is for the process with the shortest waiting time (from recent arrival in "RDY").
  - If "tie", priority is given to the process with the smallest index.
  - If "RUN" is empty, a process can directly transit from "W" via "RDY" to "RUN".

a) Please fill the first 25 time units of this following Gantt Chart:
- The state of each processes (P₁, P₂, P₃, P₄).
- Which process is in the RUN state (RUN).
- How many processes are in the Ready state (RDY).

b) Calculate (in %), how much the CPU utilization is.
c) What is the average load (in %) of the RDY state?
Process State II

There exists four processes, P1(0:2.0), P2(5:4.9), P3(10:2.9), P4(15:3.3); [where Pn(A:B) means n=process number; A=starting time; B=CPU time] with this following CPU utilization table:

<table>
<thead>
<tr>
<th>(I/O Wait = 60%)</th>
<th>Multiprogramming Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>CPU busy</td>
<td>40%</td>
</tr>
<tr>
<td>CPU busy per process</td>
<td>40%</td>
</tr>
</tbody>
</table>

Please draw a "processes/time relation" chart:

![Process State II Chart]

Process State III

(See Process State II) There exists four processes, P1(0:4.0), P2(10:4.9), P3(15:2.9), P4(20:3.3); [where Pn(A:B) means n=process number; A=starting time; B=CPU time].

Process State IV

There exists four processes, A(90: 34.6), B(80: 50), C(70: 46), D(60: 28); [where X(Y:Z) means X=process; Y=I/O Wait (%); Z=CPU time] with this following CPU utilization table:

<table>
<thead>
<tr>
<th>Multiprogramming Combination (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>CPU utilization/proc A</td>
</tr>
<tr>
<td>CPU utilization/proc B</td>
</tr>
<tr>
<td>CPU utilization/proc C</td>
</tr>
<tr>
<td>CPU utilization/proc D</td>
</tr>
</tbody>
</table>

All processes terminate at the same time. Please draw a "processes/time relation" chart and calculate the starting time of all processes!
Linux Three-Level Page Table I

This following, $008\ 0200\ 8004_{(HEX)}$, is a valid 43 bit Linux Virtual Address with three level page tables: Global Directory (10 bits), Page Middle Directory (10 bits), and Page Table (10 bits).

a) Convert the base-16 address above into base-2.
b) Complete the diagram above with its table names, indexes (in base-16), pointers (in arrow form), and memory contents (whatever/random). You may use dotes "..." for "and so on".
c) What is the size of a memory frame?

![Diagram of Linux Three-Level Page Table I]

Linux Three-Level Page Table II

What if the address is $004\ 0100\ 4002_{(HEX)}$?

Linux Three-Level Page Table III

What if the address is $000\ 0000\ 0000_{(HEX)}$?
Buddy Algorithm I

Basically, the "Buddy Algorithm" allocates pages in the power-of-2. The request will be rounded up to the next highest power of 2. Give a simple illustration of the this algorithm. Suppose, there exists a single contiguous memory of 64 pages.

a) Process A requests 7 pages.
b) Process B requests 3 pages.
c) Process C requests 9 pages.
d) Process B returns its request.
e) Process D requests 9 pages.

Buddy Algorithm II

What if there exists a single contiguous memory of 64 pages.

a) Process A requests 9 pages.
b) Process B requests 7 pages.
c) Process C requests 3 pages.
d) Process B returns its request.
e) Process D requests 1 page.
HardDisk (I/O)

- The disk rotates at 6000 RPM.
- Each track holds 1000 sectors @ 10 kbytes.
- Whenever a buffer (10 kbytes) is empty, the system will refill it at a constant rate of 5 Mbytes per second.
- Whenever a buffer is full, it will be written (emptied) to a destined sector.
- Except the rotational latency delay, ignore all other delays like buffer switch time, seek time, etc.
- For the "best case", the filled buffer will be directly written to the destined sector.
- For the "worst case", the filled buffer will have wait one disk rotation before it can be written to the destined sector.

a) **BEST CASE**: For a maximal effective transfer rate, at least how many buffers are needed? How much will be that effective transfer rate? Explain!

b) **WORST CASE**: For a maximal effective transfer rate, at least how many buffers are needed? How much will be that effective transfer rate? Explain!

### Disk Partitions I

<table>
<thead>
<tr>
<th>Device</th>
<th>----first----</th>
<th>--geom/last--</th>
<th>------sectors------</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/c0d1</td>
<td>Cyl Head Sec</td>
<td>Cyl Head Sec</td>
<td>Base</td>
</tr>
<tr>
<td></td>
<td>32 16 63</td>
<td>31 15 62</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Num</th>
<th>Sort</th>
<th>Type</th>
<th>Cyl</th>
<th>Head</th>
<th>Sec</th>
<th>Base</th>
<th>Size</th>
<th>Kb</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>p0</td>
<td>81 MINIX</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>63</td>
<td>16128</td>
</tr>
<tr>
<td>1</td>
<td>p1</td>
<td>81 MINIX</td>
<td>32</td>
<td>15</td>
<td>62</td>
<td>0</td>
<td>32256</td>
<td>16128</td>
</tr>
<tr>
<td>2</td>
<td>p2</td>
<td>81 MINIX</td>
<td>64</td>
<td>15</td>
<td>62</td>
<td>0</td>
<td>65520</td>
<td>32760</td>
</tr>
<tr>
<td>3</td>
<td>p3</td>
<td>81 MINIX</td>
<td>65</td>
<td>15</td>
<td>62</td>
<td>0</td>
<td>65520</td>
<td>32760</td>
</tr>
</tbody>
</table>

a) Divide a disk into four (4) main partitions. The first partition size is 2048 kbytes, the second one is 4096 kbytes, and the third one is 8192 kbytes. Please fill the blanks of the scheme above.

b) Why does the first partition not start from track #0?

### Disk Partitions II

Please fill the blanks of this following scheme:

<table>
<thead>
<tr>
<th>Device</th>
<th>----first----</th>
<th>--geom/last--</th>
<th>------sectors------</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/c0d1</td>
<td>Cyl Head Sec</td>
<td>Cyl Head Sec</td>
<td>Base</td>
</tr>
<tr>
<td></td>
<td>65 16 63</td>
<td>64 15 62</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Num</th>
<th>Sort</th>
<th>Type</th>
<th>Cyl</th>
<th>Head</th>
<th>Sec</th>
<th>Base</th>
<th>Size</th>
<th>Kb</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>p0</td>
<td>81 MINIX</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>63</td>
<td>16128</td>
</tr>
<tr>
<td>1</td>
<td>p1</td>
<td>81 MINIX</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4221</td>
<td>16128</td>
</tr>
<tr>
<td>2</td>
<td>p2</td>
<td>81 MINIX</td>
<td>12</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>12411</td>
<td>16128</td>
</tr>
<tr>
<td>3</td>
<td>p3</td>
<td>81 MINIX</td>
<td>28</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>28971</td>
<td>16128</td>
</tr>
</tbody>
</table>
File System

This file system is using an inode (unix) alike allocation method. The pointer size is 4 bytes. Supposed there are 12 pointers in the i-node. The first 10 ones point to "direct blocks", i.e. the content (data) of the file. The next one points to a single "single indirect block", which points to "direct blocks". The last one points to a single "double indirect block", which points to "single indirect blocks".

a) If the block size is 100 bytes, what will be the maximum size of the file?

b) If the block size is 1000 bytes, what will be the maximum size of the file?

c) If the block size is N bytes, what will be the maximum size of the file?
Fork I

```c
#include <stdio.h>
#include <stdlib.h>

main()
{
    int pid1, pid2, pid3, pid4;

    pid1=(int) getpid();    /* what is my PID ? */
    pid2=(int) fork();      /* split parent and child */
    wait (NULL);            /* parent wait for its child */
    pid3=(int) fork();      /* parent wait for its child */
    wait (NULL);
    pid4=(int) getpid();    /* parent wait for its child */
    printf("[%4d] [%4d] [%4d] [%4d]\n", pid1, pid2, pid3, pid4);
}
```

Suppose the process ID (PID) is 5000. Assume that each new child process will have the next sequential PID that is available (5001, 5002, etc.). Please write down the output of these processes!

Fork II

Please write down the output of this following C-program "isengfork1"!

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

main()
{
    int ii=0;
    if (fork() > 0) ii++;
    wait(NULL);
    if (fork() == 0) ii++;
    wait(NULL);
    if (fork() < 0) ii++;
    wait(NULL);
    printf("Result = %3.3d \n", ii);
}
```
Fork III

Please write down the output of this following C-program "isengfork1"!

```
01 /* isengfork2 (c)2008 Rahmat M. Samik-Ibrahim, GPL-like */
02 */*********************************************************************************/
04 #include <sys/types.h>
05 #include <stdio.h>
06 #include <unistd.h>
08 main()
09 {
10    int ii=2;
11    if (fork() > 0) ii--;
12    wait(NULL);
13    if (fork() == 0) ii--;
14    wait(NULL);
15    if (fork() < 0) ii--;
16    wait(NULL);
17    printf("Result = %3.3d \n",ii);
18 }
19 /*********************************************************************************/
```

Fork IV

```
01 /* cascafork (c) 2008 Rahmat M. Samik-Ibrahim, GPL-like */
02 /*------------------------------------------------------------------------*/
03 #include <sys/types.h>
04 #include <sys/wait.h>
05 #include <stdio.h>
06 #include <stdlib.h>
07 #include <unistd.h>
08 #define DISPLAY "This is PID[%5.5d]\n"
09 /*------------------------------------------------------------------------*/
10 main(void) {
11    if (fork() != (pid_t) 0) {
12       sleep(1);
13       if (fork() == (pid_t) 0) {
14          sleep(1);
15          if (fork() != (pid_t) 0) {
16             sleep(1);
17             if (fork() == (pid_t) 0) {
18                sleep(1);
19             }
20          }
21     }
22    }
23    printf(DISPLAY, (int) getpid());
24    waitpid(-1,NULL,0);
25    waitpid(-1,NULL,0);
26    exit (0);
27 }
28 /*------------------------------------------------------------------------*/
```

a) Suppose the process ID (PID) is 5000. Assume that each new child process will have the next sequential PID that is available (5001, 5002, etc.). Please write down the output sequences of these processes!

b) What happen if we delete line 12, 14, 16, and 18 [sleep()]?

c) What happen if we delete line 24 and 25 [waitpid(-1, NULL, 0) ]?
MultiThreads

```java
009 // MultiThreads (c)2006 Rahmat M. Samik-Ibrahim, GPL-like //
010 // ******************************************* MultiThreads *** //
011 public class MultiThreads {
012     public static void main(String args[]) {
013         Engine engine = new Engine(THREAD_COUNT);
014         Thread[] player = new Thread[THREAD_COUNT];
015         for (int ii=0; ii<THREAD_COUNT ; ii++) {
016             player[ii] = new Thread(new Player(ii,engine));
017             player[ii].start();
018         }
019     }
020     private static final int THREAD_COUNT = 4;
021 }
022 // ******************************************* Player *** //
023 class Player implements Runnable {
024     Player(int count, Engine eng) {
025         engine = eng;
026         player_count = count;
027     }
028     public void run() {
029         engine.play(player_count);
030     }
031     private Engine engine;
032     private int player_count;
033 }
034 // ******************************************* Engine *** //
035 class Engine {
036     public Engine(int count) {
037         idx   = count-1;
038         control = new Semaphore[count];
039         for (int ii=0; ii<count ; ii++) {
040             control[ii] = new Semaphore();
041         }
042     }
043     public void play(int ii) {
044         if (ii < idx) {
045             control[ii+1].acquire();
046         }
047         System.out.println("Player " + ii + " is up...");
048         control[ii].release();
049     }
050     private int idx;
051     private Semaphore[] control;
052 }
053 // ******************************************* Semaphore *** //
054 class Semaphore {
055     public Semaphore() {
056         value = 0;
057     }
058     public synchronized void acquire() {
059         while (value == 0) {
060             try { wait(); } 
061             catch (InterruptedException e) { }
062         }
063         value--;
064     }
065     public synchronized void release() {
066         value++;
067         notify();
068     }
069     private int value;
070 }
```

a) Please write down the output of this java program!
b) Please explain briefly, the purpose of using the semaphores in this java program!
c) Please, slightly modify the "Engine class" so that the output sequence will be the opposite of point (a). (Hint: 3 lines only, lah! :-).
Synchronization I

a) How many semaphore objects are used in this following Java program? Name them one by one!
b) Write down the output of the Java program!

```java
public class Sakit {
    public static void main(String args[]) {
        Engine engine = new Engine(strings, strseq);
        Thread[] printer = new Thread[strings.length];
        for (int ii = 0; ii < strings.length; ii++) {
            printer[ii]=new Thread(new Printer(ii, engine));
            printer[ii].start();
        }
        for (int ii = 0; ii < strings.length; ii++) {
            semaphore[ii] = new Semaphore();
        }
        sequence = 0;
        semaphore[strseq[sequence]].release();
    }
}

class Engine {
    Engine(String str[],int strseq[]) {
        this.str = str;
        this.strseq = strseq;
        semaphore = new Semaphore[str.length];
        for (int ii=0; ii<str.length; ii++) {
            semaphore[ii] = new Semaphore();
        }
        sequence = 0;
        semaphore[strseq[sequence]].release();
    }
    public void go(int ii) {
        semaphore[ii].acquire();
        System.out.print(str[ii] + "");
        if (sequence < strseq.length)
            semaphore[strseq[sequence]].release();
        else
            System.out.println();
    }
    private Semaphore[] semaphore;
    private String str[];
    private int strseq[];
    private int sequence;
}

class Printer implements Runnable {
    Printer(int ii, Engine ee) {
        number = ii;
        engine = ee;
    }
    public void run() {
        engine.go(number);
    }
    private int number;
    private Engine engine;
```
058 /************************************************************/  
059 class Semaphore {  
060    public Semaphore() { value = 0; }  
061    public Semaphore(int v) { value = v; }  
062    public synchronized void acquire() {  
063        while (value == 0) {  
064            try { wait(); }  
065            catch (InterruptedException e) { }  
066        }  
067        value--;  
068    }  
069    public synchronized void release() {  
070        value++;  
071        notify();  
072    }  
073    private int value;  
074  }

Synchronization II

001 /**************************************************************/  
002 /* MultiStrings (c) 2008 Rahmat M. Samik-Ibrahim, GPL-like */  
003 /* $Date: 2008/06/25 12:12:30 $ $Revision: 1.1 $ */  
004 /**************************************************************/  
005  
006 public class MultiStrings {  
007    public static void main(String args[]) {  
008        Engine engine = new Engine(strings, strseq);  
009        Thread[] printer = new Thread[strings.length];  
010        for (int ii = 0; ii < strings.length; ii++) {  
011            printer[ii]=new Thread(new Printer(ii, engine));  
012            printer[ii].start();  
013        }  
014    }  
015    private final static String strings[] = { "a", "an", "and",  
016        "as", "Design", "Extended", "Implementation", "Machine",  
017        "Manager", "Operating", "Resource", "Systems", "The"};  
018    private final static int[] string_array][] =  
019        {{12,9,11,3,1,5,7}, {12,9,11,3,0,10,8}, {9,11,4,2,6}};  
020    /* NameA=0 NameC=1 NameE=2  
021        NameB=0 NameD=1 NameF=2 */  
022    private final static int strseq[] = string_array[VALUE];  
023  }  
024  
025  
026 class Engine {  
027    Engine(String str[],int strseq[]) {  
028        this.str = str;  
029        this.strseq = strseq;  
030        semaphore = new Semaphore[str.length];  
031        for (int ii=0; ii<str.length; ii++) {  
032            semaphore[ii] = new Semaphore();  
033        }  
034        display = true;  
035        sequence = 0;  
036        semaphore[strseq[sequence++]].release();  
037    }
public void go(int ii) {
  semaphore[ii].acquire();
  if (display) {
    System.out.print(str[ii] + " ");
    if (sequence < strseq.length) {
      semaphore[strseq[sequence++]].release();
    } else {
      System.out.println();
      display = false;
      for (int jj=0;jj<str.length;jj++) {
        semaphore[jj].release();
      }
    }
  }
}

private Semaphore[] semaphore;
private String str[];
private int strseq[];
private int sequence;
private boolean display;

/**************************************************************/
class Printer implements Runnable {
  Printer(int ii, Engine ee) {
    number = ii;
    engine = ee;
  }
  public void run() {
    engine.go(number);
  }
  private int number;
  private Engine engine;
}

/**************************************************************/
class Semaphore {
  public Semaphore() { value = 0; }
  public Semaphore(int v) { value = v; }
  public synchronized void acquire() {
    while (value == 0) {
      try { wait(); }
      catch (InterruptedException e) { } }
    value--;
  }
  public synchronized void release() {
    value++;
    notify();
  }
  private int value;
}

a) See line 19 for the "VALUE" of line 21. What is the output of this program?
b) What is the value of "strseq.length" in line 42?
c) What is the value of "str.length" in line 47?
d) What is the purpose of the loop in line (47 to 49)? What happen if we delete those lines?