MIDTERM EXAMINATION
Fall 2012
CS604- Operating Systems

The given code is as following;

```
boolean flag[2];
int turn;
do
{
flag[i]=true;
turn=j;
while(flag[j] && turn==j);
critical section
flag[i]=false;
remainder section
} while(1)
```

Explain if the given code satisfies Mutual Exclusion or not. Justify your answer. (5)

Answer:- (Page 105)

No, it does not satisfy Mutual Exclusion because too prove mutual exclusion, note that P_i enters its critical section only if either flag[j]=false or turn=i. Also, if both processes were executing in their critical sections at the same time, then flag[0]= = flag[1]= = true. These two observations suggest that P_0 and P_1 could not have found both conditions in the while statement true at the same time, since the value of 'turn' can either be 0 or 1. Hence only one process say P_0 must have successfully exited the while statement. Hence mutual exclusion is preserved.

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

3. write down the software based solution for critical section problem (2)
Answer:- (Page 101)

do
{
Entry section
critical section
Exit section
remainder section
} while(1)

5. while working on linux, by which command you can show information about a process (2)
Answer:- (Page 66)
ps gives a snapshot of the current processes. Without options, ps prints information about processes owned by the user.

MIDTERM EXAMINATION
Fall 2012
CS604- Operating Systems

1) Difference between "progress" and " bounded time: in critical section. 2 marks
Answer:- (Page 98)
Progress:-If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder section can participate in the decision on which will enter its critical section next, and this selection cannot be postponed indefinitely.

Bounded Waiting:- There exists a bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

3) If process in background then we want its move in foreground then what unix linux command is use to moving. 3 marks
Answer:- (Page 65)
Moving a process into foreground
You can use the fg command to resume the execution of a suspended job in the foreground or move a background job into the foreground. Here is the syntax of the command.
fg [%job_id]
Where, job_id is the job ID (not process ID) of the suspended or background process. If %job_id is omitted, the current job is assumed.

خدا کے سوا کسی سے امید مت رکھو

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

4) How The open source help us to test the algorithm 3 marks

Answer:- (Page 94)

The Open Source software licensing has made it possible for us to test various algorithms by implementing them in the Linux kernel and measuring their true performance.

MIDTERM EXAMINATION
Fall 2012
CS604- Operating Systems

what is difference b/w preemptive and non-preemptive (2)

Answer:- Click here for detatail

Preemptive scheduling allows a process to be interrupted in the middle of its execution, taking the CPU away and allocating it to another process. Non Preemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

difference b/w progress and bound waiting

Answer:- Rep

MIDTERM EXAMINATION
Fall 2012
CS604- Operating Systems

Question No 1:-

Preemptive Short Job First scheduling algorithm is best algorithm for minimizing the waiting time for the process. How can you calculate the average time in preemptive Short Job First scheduling algorithm?

Answer:- (Page 80)

Example

Process	Burst time	Arrival
P_1	24	0
P_2	3	0
P_3	3	0

Gantt chart: Order P_1, P_2, P_3

Average waiting time: (0+24+27)/3 = 17

بری صحبت سے تنہائی بہتر ہے اور تنہائی سے نیک صحبت بہتر ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question No 2:-
Critical section has hardware based or software based solution. You have to write the structure of the software base solution for critical section problem?
Answer:- Rep

Question No 3:-
You may imagine the multi level feedback queue (MLFQ) scheduling algorithm is same as Short Job First scheduling algorithm. Justify your answer either yes or no?

Question No 4:-
Do you feel that reason for mknod system call and mkfifo library call failure are both same? Give reason to support your answer?
Answer:- (Page 57)
In fact, the normal file I/O system calls (close(), read(), write(), unlink(), etc.) all works with FIFOs. Since mkfifo() invokes the mknod() system call, the reasons for its failure are pretty much the same as for the mknod() call given above.

Question No 5:-
You have to explain the working of Semaphore Algorithm?
Answer:- (Page 108)
Hardware solutions to synchronization problems are not easy to generalize to more complex problems. To overcome this difficulty we can use a synchronization tool called a semaphore. A **semaphore** S is an integer variable that, apart from initialization is accessible only through two standard atomic operations: wait and signal. These operations were originally termed P (for wait) and V (for signal). The classical definitions of wait and signal are:

```
wait(S) {
while(S<=0)
;// no op
S--;
}
signal(S) {
S++;
}
```

Modifications to the integer value of the semaphore in the wait and signal operations must be executed indivisibly. That is, when one process is updating the value of a semaphore, other processes cannot simultaneously modify that same semaphore value. In addition, in the case of the wait(S), the testing of the integer value of S (S<=0) and its possible modification (S--) must also be executed without interruption.

We can use semaphores to deal with the n-process critical section problem. The n processes share a semaphore, **mutex** (standing for mutual exclusion) initialized to 1. Each process P_i is organized as follows:

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

```
do
{
wait(mutex);
Critical section
signal(mutex);
Remainder section
} while(1);
```

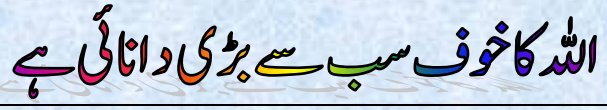
Question No 6:-
Write difference between preemptive and non preemptive scheduling algorithm? Give one name each for preemptive and non preemptive scheduling algorithm?
Answer:- Click here for detatail
Preemptive scheduling allows a process to be interrupted in the middle of its execution, taking the CPU away and allocating it to another process. Non Preemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.
FCFS is a non-preemptive scheduling algorithm.
The SJF algorithm may either be preemptive or non-preemptive. Preemptive SJF scheduling is sometimes called **shortest-remaining-time-first** scheduling.

MIDTERM EXAMINATION
Fall 2012
CS604- Operating Systems

1) Similarities between process and thread execution? 2 marks
Answer:- Click here for detail
Similarities
1) Share cpu.
2)sequential execution
3)create child
4) If one thread is blocked then the next will be start to run like process.

Dissimilarities:
1) Threads are not independent like process.
2) All threads can access every address in the task unlike process.
3)threads are design to assist one another and process might or not might be assisted on one another

2)SJF preemptive average waiting time ?2 marks
Answer:- Rep



Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

3) Define any three queues that are used in multifeedback scheduling?

4) Process synchronization of concurrent process or thread for shared data and shared resource? 3 marks

Answer:- (Page 95)

Concurrent processes or threads often need access to shared data and shared resources. If there is no controlled access to shared data, it is often possible to obtain an inconsistent state of this data. Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes, and hence various process synchronization methods are used.

5) Define data race detection is static and dynamic if it is define in detail? marks 5

Answer:- Click here for detail

Data race detection can be broadly classified as either static or dynamic. Static data race detectors [5, 28] analyze the program source code without executing it. They scale to large code bases, providing results in a short amount of time. For instance, RELAY can analyze and report races in the Linux kernel (4 MLOC) in around 5 hours. Static race detectors typically have fewer false negatives (i.e., do not miss real races) than dynamic race detectors [23]. However, static race detectors tend to have many false positives (i.e., produce reports that do not actually correspond to real races). For example, 84% of the races reported by RELAY [28] are not true races.

Dynamic data race detectors typically do not have false positives, but only detect data races in executions they can directly observe; therefore they have many false negatives. They also have high runtime overhead (e.g., 200× in the case of Intel ThreadChecker [11] and up to 8.5× in the case of FastTrack [6]), because they typically need to monitor all memory accesses and synchronization primitives. Detectors that employ sampling [13] decrease runtime overhead at the expense of introducing both false positives and false negatives. High runtime overhead indirectly decreases the coverage of dynamic detectors: they cannot be enabled in production, so they are only used by developers to detect races in executions of the program's test suite.

6) Priority scheduling algorithm is indefinite blocking or starvation define it why? marks 5

Answer:- (Page 83)

A major problem with priority- scheduling algorithms is **indefinite blocking** (or **starvation**). A process that is ready to run but lacking the CPU can be considered blocked-waiting for the CPU. A priority-scheduling algorithm can leave some low priority processes waiting indefinitely for the CPU. Legend has it that when they were phasing out IBM 7094 at MIT in 1973, they found a process stuck in the ready queue since 1967!

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

MIDTERM EXAMINATION
Fall 2012
CS604- Operating Systems

Q.2.Diff b/w bounded and progress in respect to critical section?
Answer:- Rep

Q.3.Explain Semaphore Algorithm?
Answer:- Rep

Q.4: For terminating a process which command you will use ?
Answer:- (Page 69)
You can terminate a foreground process by pressing <Ctrl-C>.
You can also terminate a process with the kill command. When executed, this command sends a signal to the process whose process ID is specified in the command line.
Here is the syntax of the command.
kill [-signal] PID

Q.5: Every operating system creates an identified operation or or the process.which includes (data copying, the code ,heap , When UNIX fork () system cal is make at the kernel level as well as at the User level ?
Answer:- (Page 73)
Support for threads may be provided at either user level for *user threads* or by kernel for *kernel threads.*

User threads are supported above kernel and are implemented by a thread library at the user level. The library provides support for thread creation, scheduling, and management with no support from the kernel. Since the kernel is unaware of user-level threads, all thread creation and scheduling are done in the user space without the need for kernel intervention, and therefore are fast to create and manage. If the kernel is single threaded, then any user level thread performing a blocking system call will cause the entire process to block, even if other threads are available to run within the application. User thread libraries include POSIX Pthreads , Solaris 2 UI-threads, and Mach Cthreads.

Kernel threads are supported directly by the operating system. The kernel performs the scheduling, creation, and management in kernel space; the kernel level threads are hence slower to create and manage, compared to user level threads. However since the kernel is managing threads, if a thread performs a blocking system call, the kernel can schedule another thread in the application for execution. Windows NT, Windowss 2000, Solaris, BeOS and Tru64 UNIX support kernel threads.

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Q:6 Diff b/w FIFO and Pipe ?

Answer:- Click here for detail

A FIFO is similar to a pipe. A FIFO (First In First Out) is a one-way flow of data. FIFOs have a name, so unrelated processes can share the FIFO. FIFO is a named pipe. difference between pipes and FIFOs is the manner in which they are created and opened. Once these tasks have been accomplished, I/O on pipes and FIFOs has exactly the same semantics.

The difference between fifos and pipes is that the former is identified in the file system with a name, while the latter is not. That is, fifos are named pipes. Fifos are identified by an access point which is a file within the file system, whereas pipes are identified by an access point which is simply an allotted inode. Another major difference between fifos and pipes is that fifos last throughout the life-cycle of the system, while pipes last only during the life-cycle of the process in which they were created. To make it more clear, fifos exist beyond the life of the process. Since they are identified by the file system, they remain in the hierarchy until explicitly removed using unlink, but pipes are inherited only by related processes, that is, processes which are descendants of a single process.

MIDTERM EXAMINATION
Fall 2012
CS604- Operating Systems

Q#3: if processor is run in the background we want to move in foreground write the LINIX/UNIX command for this process?(3 M)

Answer:- Rep

Q#5: Semaphore is variable and abstract data type that provide a simple but useful abstraction for controlling access by multiple excess to common resource in parallel environment A semaphore has record how many units are particular source are available, coupled with operations to safely adjust the records as units are required or became free and its necessary waits until a unit of the resources becomes available. You have to identify the drawbacks which are due to using semaphore?(5M)

Answer:- (Page 109)

The main disadvantage of the semaphore discussed in the previous section is that it requires **busy waiting**. While a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code. This continual looping is clearly a problem in a real multiprogramming system, where a single CPU is shared among many processes. Busy waiting wastes CPU cycles that some other process may be able to use productively. This type of semaphore is also called a **spinlock** (because the process spins while waiting for the lock). Spinlocks are useful in multiprocessor systems. The advantage of a spinlock is that no context switch is required when a process must wait on a lock, and a context switch may take considerable time. This is, spinlocks are useful when they are expected to be held for short times. The definition of semaphore should be modified to eliminate busy waiting.

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Q#6: Out of (SRTF) and round robin scheduling algorithm which one is best suited to be used in Time-sharing-system where response time is an important performance criteria? Give reasons in your answer?(5M)

Answer:- (Page 87)

Preemptive SJF scheduling is sometimes called **shortest-remaining-time-first** scheduling.

Typically, round robin has a higher average turnaround than SJF, but better response. In timesharing systems, shorter response time for a process is more important than shorter turnaround time for the process. Thus, round-robin scheduler matches the requirements of time-sharing systems better than the SJF algorithm. SJF scheduler is better suited for batch systems, in which minimizing the turnaround time is the main criterion.

MIDTERM EXAMINATION
Spring 2012
CS604- Operating Systems

1disadvantages of One to one threads

Answer:- (Page 71)

The main disadvantage of this model is the overhead of creating a kernel thread per user thread.

2. Assume

i)that each process executes at a nonzero speed

ii)No assumption can be made regarding the relative speeds of the N processes.

These two assumptions can be used for ? (2 marks)

Answer:- (Page 98)

These are used for Solution to the Critical Section Problem.

3 a job suspended or running in back ground to move it to the fore ground which command is used?

Answer:- Rep

5 when a process makes its copy which things are that must be copy to new new process(5)

Answer:- (Page 37)

When the fork system call is executed, a new process is created. The original process is called the parent process whereas the process is called the child process. The new process consists of a copy of the address space of the parent. This mechanism allows the parent process to communicate easily with the child process. On success, both processes continue execution at the instruction after the fork call, with one difference, the return code for the fork system call is zero for the child process, while the process identifier of the child is returned to the parent process. On failure, a -1 will be returned in the parent's context, no child process will be created, and an error number will be set appropriately.

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

MIDTERM EXAMINATION
Spring 2012
CS604- Operating Systems

Q.21: differentiate Entry Section and Reminder Section. (2 Marks)

Answer:- (Page 97)

When a process executes code that manipulates shared data (or resource), we say that the process is in its critical section (for that shared data). The execution of critical sections must be mutually exclusive: at any time, only one process is allowed to execute in its critical section (even with multiple processors). So each process must first request permission to enter its critical section. The section of code implementing this request is called the **entry section**. The remaining code is the **remainder section**.

Q.23: UNIX System V scheduling algorithm (3 marks)

Answer:- (Page 90)

UNIX System V scheduling algorithm is essentially a multilevel feedback priority queues algorithm with round robin within each queue, the quantum being equal to1 second. The priorities are divided into two groups/bands:

- Kernel Group
- User Group

اپنی مرضی اور اللہ کی مرضی میں فرق کا نام غم ہے

اس سے پہلے کہ تمہیں شہوت فتنے میں ڈالے نکاح کر لو

وہ لوگ مبارک ہیں جو الفاظ سے نصیحت نہیں کرتے بلکہ عمل سے کرتے ہیں

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari