Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University

Midterm Exam
Operating Systems – Spring 2017

Instructions

1. The point values for each question are marked in square brackets.
2. The exam time is 2.5 hours (9:50 – 11:50 Beiing time).
3. The total possible points in the exam are 160.
4. Please answer in English on a separate piece of paper. After the exam, you can snapshot using a camera or scan it, and submit on learn.tsinghua.edu.cn as homework "midterm1".
5. The exam is open book and open notes. However, you cannot talk to other people or search online during the exam. All questions must be directed to the administrating TAs over the WeChat Group or GoToTraining channels – Please let everybody to see it. Private communications with the TAs are prohibited.
6. You understand that a violation on item 5 will result in an immediate reporting to the institute and education affairs office for disciplinary action, which will cause a Fail grade in this course, and may lead to the termination of your student status.

Don't Panic

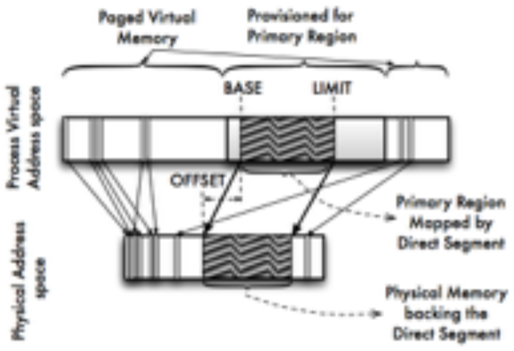
Good Luck!

1. *1 [10]* Direct Segment and Memory Management

Researchers propose an abstraction called direct segment. In a paper in ISCA'13, Basu et.al. explained that

"we translate a contiguous virtual address range directly onto a contiguous physical address range through hardware support called a direct segment …… This contiguous virtual address range can be arbitrarily large (limited only by the physical memory size of the system) and is mapped using a small fixed-sized hardware. Any virtual address outside the aforementioned virtual address range is mapped through conventional paging. "

And the following is a figure reproduced from the above paper explaining how direct segment works.


Paged Virtual Memory
Provisioned for Primary Region
BASE
LIMIT
Process Virtual Address space
OFFSET
Primary Region Mapped by Direct Segment
Physical Address space
Physical Memory backing the Direct Segment

Can you give at least two reasons why people propose this change, what do we gain from the change? What is the potential problems with this change? What kind of applications (which software on which hardware) could benefit the most from this change?

2 [10]

A Java ReentrantLock is owned by the thread last successfully locking, but not yet unlocking it. A thread invoking lock() will return, successfully acquiring the lock, when the lock is not owned by another thread. The method will return immediately if the current thread already owns the lock (the unlock only need to happen once no matter how many times a thread locks it).

Please use the Semaphore P() and V() primitive to implement an reentrant lock. You can obtain the thread id with the Java API Thread.currentThread().getid()

```
class ReentrantLock {
  void Lock(){

  }
  void Unlock(){

  }
}
```

3 [20] Suppose we replace the wait() and signal() operations of monitors with a single construct await(B), where B is a general Boolean expression that causes the process executing it to wait until B becomes true.
a. Please use this construct to rewrite the reader / writer code shown in lecture.
b. Explain why, in general, this construct cannot be implemented efficiently.
c. What restrictions can we put on the await statement so that it can be implemented efficiently?

4[10]
/my/b is a link to a normal text file /your/a of 2KB in size. How many file system blocks (4KB) will change (update / create / remove etc) after the following operations? Answer the above question for both symbolic links and hard links. In the sed case, the edit actually happens in the file (i.e. there is at least a match for the pattern in the file). You can get partial credit if you list what blocks changes.

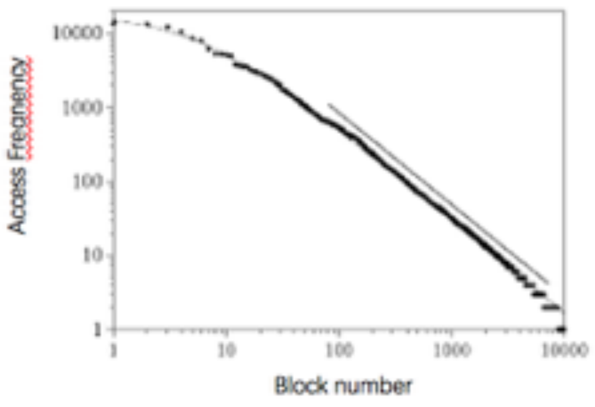
	Hard link	Symbolic link
Create the link /my/b	(ln /your/a /my/b)	(ln –s /your/a /my/b)
rm /my/b		
`sed "s/a/b/g" /my/b`		
`sed "s/a/b/g" /your/a`		
mv /your/c /your/a `sed "s/a/b/g" /my/b`		

5[20]
Consider that you have 1M (1M = 1024*1024 = 2^{20}) file system blocks of 4KB each. You are making 100M requests to these blocks. Assuming that the service time of the disk follows an exponential distribution with a mean of 10ms. Please calculate the average response time under the following conditions. Please show how you get the answer for partial credit.

1) The requests to the blocks are uniformly random arriving with an exponentially distributed inter-arrival time with mean of 10ms. There is no cache anywhere in the process. The requests are served in a FIFO order.

2) You have a 2GB memory cache with your choice of page replacement policies. The cache access time is zero (whether it is a hit or a miss).

3) The requests to the pages are not uniform, and some blocks are accessed significantly more often than others. Specifically, the figure below plots the frequency of the top 10000 most frequently accessed pages (note that both axis are in log scale). You still have a 2GB of cache memory with your choice of replacement policy.



4) If we do NOT have to serve requests in a FIFO order. Can you design a better page replacement policy to improve the average response time? You only need to describe the policy and provide a qualitative argument why it works better. No calculation is required.

6[20]

Consider a machine with a physical memory of 8 GB, a page size of 8 KB, and a page table entry size of 4 bytes.

a. How many levels of page tables would be required to map a 46-bit virtual address space if every page table entry fits into a single page? Be explicit in your explanation.

b. How much physical memory is needed for a process with three pages of virtual memory (for example, one code, one data, and one stack page)?

c. Assume the physical memory address is 34 bits. You have 4096 cache blocks. Each block in the cache holds 32 bytes of data. In order to address a specific byte of data, you will have to split the address into the cache tag, cache index and byte select. If you have a 4 way associative cache, how long will each component be (in bits) and why?

d. Assume that you have a fully associative TLB. At least how many bits do you need per TLB entry? What are these bits? The size of virtual and physical address space is defined in previous questions.

7 [10]

Consider a Unix-like file system with the inode structure the same as the one we discussed in the lecture. The file system has a buffer that can only hold 2 blocks in memory, including both data and metadata blocks (but it does not include runtime information like the open file table etc.). For the following sequence of operations, how many disk IOs does one need to access the following blocks? The system has a clean start (i.e. before step 1, the cache and all meta data is empty). *Note that we might not provide full information here for a definite answer, but you will need to state any additional assumptions you make with your answer.*

1) Accessing block 0 in file /usr/home/os/lecture1.mp4;
2) Accessing block 820 in the same file

3) Accessing block 821 in the same file
4) Accessing block 0 in file /usr/home/os/lecture2.mp4

8 [30]

Consider the following three processes with periodic CPU bursts (other than the bursts, they are blocked).

P1: burst 5ms then block for 120ms
P2: burst 40ms then block for 40ms
P3: burst 1ms then block for 249ms
P4: infinite burst time (takes as much CPU it receives)

All four processes start from time zero. For the following questions, for simplicity, If there are is any case that two processes arrives at exactly the same time and the policy does not choose one, you can use the following priority: P1>P2>P3>P4.

1) Consider a Round-Robin (RR) scheduler with quantum=20ms. Please plot the Gantt Chart for the tasks running between 0 and 300ms. What is the average response time for each burst for P1 and P3?

2) Plot the same Gantt Chart, but using Shortest Remaining Time First (SRTF) scheduler (recall from the lecture: we are scheduling for CPU bursts here). What is the average response time for each burst for P1 and P3? Is SRTF a perfect solution? What problems can it cause?

3) Consider the linux Complete Fair Scheduler (CFS). Plot the Gantt Graph using this scheduler given T_{min}=4ms and T_l=20ms.

```
Q: set of runnable processes, initially empty
runtime[p]: total running time of process p
$T_{min}$, $T_l$: parameters of the scheduler

Procedure NewRunnableProcess(p)
    runtime[p] ← max(runtime[p], min{ runtime[p'] | p' in Q })
    Add process p to Q
End

Procedure FinishRunning(p)
    runtime[p] ← runtime[p] + running time of process p
    Add process p to Q if p is still runnable
End

Procedure ScheduleNext()
    p ← argmin{ runtime[p'] | p' in Q }
    Schedule process p to run with max running time max($T_l$ / |Q|, $T_{min}$)
    Remove process p from Q
End
```

4) We realize that each application process knows its own burst pattern, and how soon it wants the burst to be handled (called a deadline). Can you design a scheduler that is fair, and yet allow processes with small and infrequent CPU bursts, such as P3, to enjoy a small average response time? *(Hint: though it is not the only solution, you can consider the priority donation case in your lottery scheduling project)*.

9[10]

Consider the following resource-allocation policy. Requests for and releases of resources are allowed at any time. If a request for resources cannot be satisfied because the resources are not available, then we check any processes that are blocked waiting for resources. If a blocked process has the desired resources, then these resources are taken away from it and are given to the requesting process. The vector of resources for which the blocked process is waiting is increased to include the resources that were taken away.

For example, a system has three resource types, and the vector Available is initialized to (4,2,2). If process P0 asks for (2,2,1), it gets them. If P1 asks for (1,0,1), it gets them. Then, if P0 asks for (0,0,1), it is blocked (resource not available). If P2 now asks for (2,0,0), it gets the available one (1,0,0), as well as one that was allocated to P0 (since P0 is blocked). P0's Allocation vector goes down to (1,2,1), and its Need vector goes up to (1,0,1).

Question: Can deadlock occur? If you answer "yes," give an example. If you answer "no," specify which necessary condition cannot occur.

10[5]

The Linux kernel has a policy that a process cannot hold a spinlock while attempting to acquire a semaphore. Explain why this policy is in place.

11 [5]

Linux man page states that

CPU affinity is a scheduler property that "bonds" a process to a given set of CPUs on the system. The Linux scheduler will honor the given CPU affinity and the process will not run on any other CPUs. Note that the Linux scheduler also supports natural CPU affinity: the scheduler attempts to keep processes on the same CPU as long as practical for performance reasons.

Explain how CPU affinity can affect performance.

11[10] True/False Questions. In the following, please *EXPLAIN* your answer in TWO SENTENCES OR FEWER (Answers longer than this may not get credit!). Also, answers without an explanation *GET NO CREDIT.*

a. To choose which block to store data to on an SSD, RANDOM is as good as other choices as there is no seek time

b. In the Nachos priority scheduler, if a HIGH priority thread is waiting for a LOW priority thread to release a lock, but there are NO OTHER THREADS in the system, the LOW priority thread's effective priority should be HIGH.

c. During a system call, the user level process should first trap to kernel, and they copy the memory address of the kernel routine to be called, as well as the arguments of the call to the kernel.

d. User level driver eliminates the need of crossing the application-kernel boundary, and thus more efficient. Thus, we should move all drivers from kernel space to user space.

e. The SEDA architecture breaks up data processing into stages connected by queues, and uses a single separate thread to handle each stage.

THE END. Congratulations!