Name: ______________________________

CS 4/53201 **Final Exam** **Operating Systems**

Wednesday 17 December 1998

1. List those items that comprise the state of a process. (5 points)

Program code, static data, dynamic data (stack & heap), registers (general-purpose registers, PC, SP, PSW), OS resources, accounting info, etc.

2. Explain how a thread differs from a process. (7 points)

A process can have multiple threads inside it. The process as a whole owns resources — address space, code, global variables, heap, and OS resources. A thread is a unit of scheduling, and it shares most resources with other threads in the same process, yet it has its own PC, SP, and stack. Finally, switching between threads is easier and faster than switching between processes.

3. A complex operating system like UNIX schedules processes at several levels. (14 points)

a. What does the medium-term scheduler (swapper) do? (3 points)

Suspends processes and swaps them out to disk whenever there isn't enough memory space for the running process.

b. What does the short-term scheduler (CPU scheduler) do? (3 points)

Picks one of the ready processes to execute on the CPU.

c. A round-robin CPU scheduler may perform a context switch 100-1000 times a second. What is a context switch, and what does the operating system have to do during one? (4 points)

A context switch is the stopping of one process and the starting of another. The OS must save the registers (PC, SP, and general-purpose registers) and any other state information in the process' PCB.

d. Following up on part (c.) above, why is a context switch performed so often? (2 points)

To provide adequate response time for interactive users.

e. Following up on part (c.) above, why isn't a context switch performed more often? (2 points)

Because a context switch takes time, and if that time is high in proportion to the time slice, too much CPU time will be spent on context switches.

Name: ______________________________

4. Consider this algorithm for mutual exclusion. (7 points)

```
Process 1 {                         Process 2 {
   while true {                        while true {
      while (turn == 1)                   while (turn == 2)
         ;  /* do nothing */                 ;  /* do nothing */
      critical section                    critical section
      turn = 2;                           turn = 1;
      other non-critical code             other non-critical code
   }                                   }
}                                   }
```

a. Does it satisfy mutual exclusion? Explain. (2 points)

Yes, a process can only enter the CS when it is its turn, and each process declares that it’s the other’s turn when if finishes the CS.

b. Does it prevent deadlock? Explain. (2 points)

Yes, one process can always go on — both will never be stuck waiting (unless turn incorrectly starts at some value other than 1 or 2).

c. What problems does this solution have? (3 points)

The main problem is that turn is explicitly passed between processes, so they have to alternate. This isn’t ideal if one process needs the CS much more often than the other. Another problem is that if a process dies in the CS, the other process will starve.

5. Consider this implementation of semaphores. (9 points)

```
wait(s):                            signal(s):
   disable interrupts                  disable interrupts
   while (s <= 0)                      s = s + 1;
      ;  /* do nothing */
   s = s – 1;
   enable interrupts                   enable interrupts
```

a. What is the purpose of disabling the interrupts? (3 points)

To make the code implementing wait and signal atomic, so that the variables in that code are always in a consistent state.

b. What problems are there with disabling the interrupts? (3 points)

1. Doesn’t work on multiprocessors. 2. Can interfere with the timer. 3. Wastes CPU time busy-waiting. 4. No queue of blocked threads. But — the main problem is that it doesn’t work — see question (c.) below.

c. What problems are there with the “do nothing” loop in wait(s)? (3 points)

While wait is busy-waiting, the OS can’t switch to signal, so signal can’t change the value of s, so wait will wait forever.

Name: ______________________________

6. **Most modern operating systems support semaphores, locks, and condition variables. (12 points)**

   a. **What are semaphores used for? (3 points)**

   Mutual exclusion and synchronization.

   b. **What are locks used for? (2 points)**

   Mutual exclusion.

   c. **What are condition variables used for? (2 points)**

   Synchronization.

   d. **Briefly describe how semaphores differ from condition variables. (5 points)**

   Semaphores can’t be nested, so they don’t allow synchronization within critical sections, but condition variables are designed to be used inside locks, so they provide that functionality. Also, semaphores have a value, while condition variables do not, so if one thread does a semaphore signal and then another one does a wait, the second will not wait, but if one thread does a condition variable signal and then another one does a wait, the second will wait.

7. **Describe the shortest-remaining-time (SRT) CPU scheduling algorithm. (7 points)**

   SRT is a preemptive version of SJF. It chooses the process with the smallest next CPU burst (predicting that time on the basis of past history) and runs that process preemptively, until either it blocks or terminates, or a new process enters the ready queue (either a new process or one previously blocked).

8. **One of the techniques for dealing with deadlock is deadlock avoidance. (9 points)**

   a. **Briefly describe the Banker’s algorithm for deadlock avoidance. (3 points)**

   The Banker considers each request for resources, and determines whether or not that request will lead to a safe state. If so, the request is granted; if not, it is postponed.

   b. **What are the advantages of deadlock avoidance over deadlock detection and recovery? (3 points)**

   No need to terminate a process or preempt resources and roll back the state.

   c. **What are the disadvantages of deadlock avoidance using the Banker’s algorithm? (3 points)**

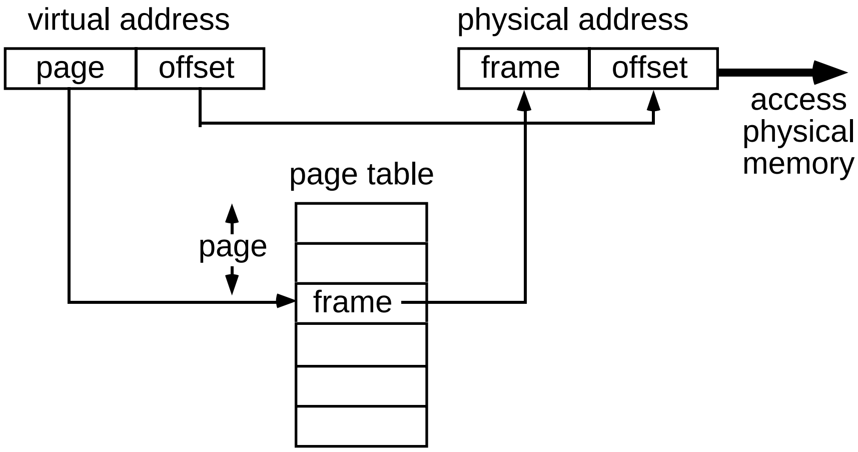
   The biggest disadvantage is the huge overhead of performing this algorithm for every resource request. Other disadvantages are: number of processes and resources must be fixed, maximum number of resources must be known in advance, and synchronization between processes isn't considered.

9. Many concepts and algorithms are applicable to more than one problem. For example, the first-fit and best-fit algorithms can be used to place objects into holes. Describe at least two of the three situations discussed in class for which these algorithms can be used in an operating system and how they might be used in each situation. For extra credit, do this for all three situations. (6 points)

Allocating memory space in the heap.
Allocating memory space when using fixed-size partitions or segmentation.
Allocating disk space when using contiguous allocation.

10. Demand paging is the most common method of memory management used in modern operating systems. (12 points)

a. Draw a diagram and explain how virtual addresses are converted into physical addresses



b. What kind of fragmentation can occur in paging? Explain. (3 points)

Internal fragmentation can occur in the last frame of a process.

b. What is a page fault, and what must the operating system do when this happens? (4 points)

A page fault occurs when a referenced page is not in virtual memory. When this happens, the OS must "page in" the page — bring it from disk into a free frame in physical memory, possibly replacing an existing page. It must also update the page table and the "present" bit for that page.

11. Fill out the following table, listing those items that are stored in each data structure in UNIX. For extra credit, do the same thing for Nachos. (12 points)

Item	UNIX	Nachos (extra credit)
Partition (4 points)	Boot block Super block Ilist Data blocks (for directories and files)	Free map Directory File headers Data blocks (for files)
File descriptor (5 points)	Type Access permission Link count Owner, group Size Access time List of data blocks	List of data blocks Number of bytes Number of sectors
Directory (3 points)	File name & inumber	File name & fileheader