University of Waterloo
Midterm Examination
Term: Winter Year: 2007

Solution

Problem 1 (10 marks)

a. **(2 marks)** What is the main difference between an exception and an interrupt?

————————begin solution————————

Exceptions are generated by executing user-level programs. Interrupts are generated by hardware devices.

————————end solution————————

b. **(2 marks)** What is the difference between context switching between two threads in the same process and context switching between two threads that are not part of the same process?

————————begin solution————————

When context switching between two threads in the same process you don't need to change anything in the MMU (e.g., base register or TLB). When switching between two threads in different processes they are in different address spaces so the MMU needs to be modified for the new address space.

————————end solution————————

c. **(2 marks)** In Nachos why are there two stacks for every thread (assuming the threads are not user-level threads)?

————————begin solution————————

While executing in privilege/kernel mode the thread needs a stack so there must be a stack in the kernels address space. While executing in user mode the thread needs a stack so there must be a stack in the process's address space.

————————end solution————————

d. **(2 marks)** In a real operating system running on a real machine would each thread require two stacks (again assume that the threads are not user-level threads)? Why or why not?

————————begin solution————————

YES.
For the same reasons provided above.

————————end solution————————

e. **(2 marks)** Assume there are three long running processes in the system P_1, P_2 and P_3. P_1 has two kernel threads, P_2 has three user-level threads and P_3 has four kernel threads. If none of the programs executes any system calls that cause them to block, there are no other processes in the system and the kernel use a pre-emptive round robin scheduling algorithm, what fraction of the CPU time will process P_1 be allocated?

————begin solution————

```
2 + 1 + 4 kernel threads = 7.
P1 has 2 of the 7 threads = 2/7.
```

————end solution————

Problem 2 (10 marks)
A program has been compiled for a system with a **page size of 1000 bytes**. We are told that all segments are **page aligned** and that the size of the various segments are as follows:

Code segment = 4354 bytes
Read-only data segment = 2478 bytes
Initialized data segment = 224 bytes
Uninitialized data segment = 501 bytes
Stack = 7000 bytes

a. **(3 marks)** Assuming that the header used in the executable file is 423 bytes, what is the size of the executable used to store this program (in bytes)? Assume that the structure of the executable file is similar to the Nachos noff file. Show your work.

————begin solution————

Code	RO	Init	Header	
4354	+ 2478	+ 224	+ 423	= 7479

————end solution————

b. **(3 marks)** How many entries will there be in the page table for this program (assuming that segments are continguous in virtual memory as done in Nachos)?

————begin solution————

The header is not loaded. Each segment ends up on separate page, hence the following usage:

Code	RO	Init	Uninit	Stack	
5	+ 3	+ 1	+ 1	+ 7	= 17

————end solution————

c. **(2 marks)** On a system with a page size of 2048 bytes there are two executable files named `PgmA` and `PgmB`. The size of the file `PgmA` is 100,234 bytes and `PgmB` is 10,290 bytes.

Which program will have the larger page table (circle the correct answer and justify in one sentence)?

```
a) PgmA   b) PgmB   c) They will be equal   d) Not enough information is provided
```

————begin solution————

```
                                                 ----------------------------------------
a) PgmA   b) PgmD   c) They will be equal   | d) Not enough information is provided |
                                                 ----------------------------------------
```

The uninitialized data can be arbitrarily large and require arbitrarily many pages, hence it is impossible to decide the number of pages used by each program.

————————end solution————————

d. **(2 marks)** On a system with a page size 4096 bytes we notice two executable files named `PgmC` and `PgmD`. The size of the file `PgmC` is 56,294 bytes and `PgmD` is 56,294 bytes.

Which program will have the larger page table (circle the correct answer and justify in one sentence)?

```
a) PgmC   b) PgmD   c) They will be equal   d) Not enough information is provided
```

————————begin solution————————

```
                                                 ----------------------------------------
a) PgmC   b) PgmD   c) They will be equal   | d) Not enough information is provided |
                                                 ----------------------------------------
```

The uninitialized data can be arbitrarily large and require arbitrarily many pages, hence it is impossible to decide the number of pages used by each program.

————————end solution————————

Problem 3 (10 marks)
Assume that Apple has released a version of their operating system (called MAC/MIPS) that executes on the MIPS processor. Also assume that we have a bunch of Macs in a lab that are running MAC/MIPS and that all of the programs compiled to execute on the MAC/MIPS system have been compiled and linked with the same version of gcc-mips and coff2noff that you have used to compile programs to execute on your version of Nachos (yes, assume the executable file formats are identical).

a. **(4 marks)** Why is a MAC/MIPS executable unlikely to execute correctly on Nachos? (Provide a few possible reasons).

————————begin solution————————

Because it is unlikely that the system call numbers and the system call parameters will be the same on both systems. It is also likely that Nachos does not support all of the system calls that are supported on MAC/MIPS.

————————end solution————————

b. **(6 marks)** Assume Apple won't give us the source code to MAC/MIPS and they won't modify their operating system, explain what information you would need from Apple and what you would modify in Nachos to allow us to execute MAC/MIPS programs on Nachos.

————————begin solution————————

```
We would need to find out:
  o the list of all of the system calls in MAC/MIPS
  o what all of the parameters are for those system calls
  o what all of the system calls do including what
    the return values are and possible error codes that
    are used
We would then need to:
  o implement all of the system calls using the same
    system call numbers, the same parameters and the
    same return codes and error conditions.
```

————————end solution————————

Problem 4 (10 marks)
The local laundromat has just entered the computer age. As each customer enters, he or she puts coins into slots at one of two stations and types in the number of washing machines he/she will need. The stations are connected to a central computer that automatically assigns available machines and outputs tokens that identify the machines to be used. The customer puts laundry into the machines and inserts each token into the machine indicated on the token. When a machine finishes its cycle, it informs the computer that it is available again. The computer maintains an array `available[NMACHINES]` whose elements are non-zero if the corresponding machines are available (`NMACHINES` is a constant indicating how many machines there are in the laundromat), and a semaphore `nfree` that indicates how many machines are available. The `available` array is initialized to all ones, and `nfree` is initialized to `NMACHINES`. The code to allocate and release machines is as follows:

```
int allocate() {  /* Returns index of available machine.*/
  int i;
  P(nfree);     /* Wait until a machine is available */
  for (i=0; i < NMACHINES; i++) {
    if (available[i] != 0) {
      available[i] = 0;
      return i;
    }
  }
}
release(int machine) { /* Releases machine */
  available[machine] = 1;
  V(nfree);
}
```

a. **(4 marks)** It seems that, if two people make requests at the two stations at the same time, they will occasionally be assigned the same machine. This has resulted in several brawls in the laundromat, and you have been called in by the owner to fix the problem. Assume that one thread handles each customer station. Explain how the same washing machine can be assigned to two different customers.

————————begin solution————————

Because P(nfree) only checks that there are 1 or more machines available two people could arrive at the same time and both could be allocated the same machine because the available[i] array is a critical section that is not protected.

————————end solution————————

b. **(6 marks)** Describe how to modify the code above to eliminate the problem. Please denote your changes directly on the code shown above.

———————begin solution———————

```
mutex is a binary semaphore with initial value = 1;
or and array of binary semaphores is used
each with initial values = 1;

int allocate() {  /* Returns index of available machine.*/
  int i;
  P(nfree);     /* Wait until a machine is available */
  for (i=0; i < NMACHINES; i++) {
    P(mutex);  or  P(mutex_array[i]);  **** ADDED
    if (available[i] != 0) {
      available[i] = 0;
      V(mutex); or V(mutex_array[i]);  **** ADDED
      return i;
    } else {                           **** ADDED
      V(mutex); or V(mutex_array[i]);  **** ADDED
    }
  }
}
release(int machine) { /* Releases machine */
  P(mutex); or P(mutex_array[machine]);  **** ADDED
    available[machine] = 1;
    V(nfree);
  V(mutex); or V(mutex_array[machine]);  **** ADDED
}
```

———————end solution———————

Problem 5 (10 marks)
Consider the following page table used for address translations:
(for this portion of the question, all numbers are expressed using base 10).

Virtual page	Frame
0	3
1	10
2	9
3	2
4	0

Assume the page size is 1024 bytes, convert the following virtual addresses into physical addresses (show your calculations):

a. **(2 marks)** 697

———————begin solution———————

697 = 0 * 1024 + 697 and so vpn = 3.
The real Address is 3 * 1024 + 697 = 3769

——————end solution——————

b. **(2 marks)** 1054

——————begin solution——————

1054 = 1024 + 30 and so is on page 1.
The real Address is 10*1024 + 30 = 10270;

——————end solution——————

c. **(2 marks)** 1024

——————begin solution——————

```
page 0 is virtual address 0 .. 1023.
So 1024 =  is vpn = 1  offset = 0
The real Address is 10 * 1024 = 10240
```

——————end solution——————

d. **(4 marks)** If possible, convert the **physical address** 2075 into a **virtual address**. If not explain why it can not be done.

——————begin solution——————

```
2075 is on physical frame 2 with offset 27.
From the page table we see that frame 2 is virtual page 3.
Virtual page 3 is 3 * 1024 = 3072 + the offset (27)
So the virtual address = 3099.
```

——————end solution——————

Problem 6 (10 marks)
Consider the following segment table (recall the prefix 0x means the number is in hexidecimal, and that each hexidecimal character represents 4 bits).

segment	base
0	0x00700000
1	0x00200000
2	0x00500000
3	0x09000000

Assume a virtual address is 32 bits and that the virtual address permits a maximum of 16 segments.

a. **(2 marks)** How many bits will be used to represent the segment number?

——————begin solution——————

$2^4 = 16$ so 4 bits.

——————end solution——————

b. **(2 marks)** How many bits will be used to represent the offset?

——————begin solution——————

$32 - 4 = 28$

——————end solution——————

c. **(2 marks)** What is the maximum size of a segment in this system in bytes (expressed as an equation).

——————begin solution——————

$2^{28} = 268,435,456 = 256MB.$

——————end solution——————

Using the segment table above and the virtual addressing scheme described above, convert the following virtual addresses (expressed in hexidecimal) into physical addresses (also expressed in hexidecimal) :

d. **(2 marks)** 0x00002092

——————begin solution——————

```
The first 4 bytes are 0 so the segment is 0.
It starts at address 0x00700000 so the physical address is:
0x00700000 + 2092 = 0x00702092
```

——————end solution——————

e. **(2 marks)** 0x30654321

——————begin solution——————

```
The first 4 bytes are 3 so the segment is 3.
It starts at address 0x09000000 so the physical address is:
0x09000000 + 654321 = 0x09654321
```

——————end solution——————

Problem 7 (10 marks)

Consider a NachOS system in which k ($k > 1$) processes are running. One process is running at low priority, and the remaining $k - 1$ processes are running at normal priority. Each process' program is described by the following pseudo-code:

```
REPEAT 5 TIMES {
    compute for C time units
    write a single character to the NachOS output console
}
```

Characters are written to the synchronous output console using the NachOS `Write` system call. The low priority process writes the character "L". The normal priority processes write the character "N". The synchronous output console outputs characters one at a time in the order in which the `Write` requests are made. Assume that that output console requires W time units to output a single character.

Assume that the NachOS scheduler is a round robin preemptive scheduler with quantum q that has been modified to understand priorities, as was required for Assignment 1. Specifically, a lower priority process will never run if there is a runnable process of higher priority.

For the purposes of this question, ignore context switching overhead.

a. **(5 marks)** Suppose that $k = 2$ and $C < W < q$. Under this assumption, what is the sequence of N's and L's that will be produced on the output console by the processes? Explain your answer.

——————begin solution——————

Because there are only two threads. One has normal priority (call this T_L) and the other has normal priority (call this T_N). T_N runs first, computes for C units of time, writes an 'N' and blocks for W units of time ($W > C$). While T_N is blocked, T_L runs, computes for C units of time, ($C < W$), writes an 'L' and blocks for W units of time ($W > C$). At this point they have printed "NL". T_N unblocks first and the sequence will be repeated 5 times, producing:

NLNLNLNLNL
that is
NL repeated 5 times.

——————end solution——————

b. **(5 marks)** Suppose that $k = 3$ and $W < C < q$. Under these assumptions, what is the sequence of N's and L's that will be produced on the output console by the processes? Explain your answer.

——————begin solution——————

Because there are 3 threads. Two have normal priority (call them T_N threads) and the other has low priority (call this thread T_L). One of the normal priority threads is running first (without loss of generality call this thread T_N1). T_N1 computes for C units of time, writes an 'N' and blocks for W units of time ($W < C < q$). While T_N1 is blocked (for W) units of time T_N2 runs, computes for C units of time, writes an 'N' and blocks for W units of time ($W < C < q$). Because T_N1 was blocked for ($W < C$) units of time, it will be unblocked and ready to run before T_N2 is finished computing. So when T_N2 blocks, T_N1 will be run. That sequence will repeat until both of these threads have finished (producing 5 N's each) and finally T_L will be able to run and it will print 'L' 5 times.

Therefore the output produced is:
NNNNNNNNNNLLLLL
that is
10 N's followed by 5 L's.

——————end solution——————

Problem 8 (10 marks)
This question uses the following notation to describe resource allocation in a computer system (as used in the course notes):

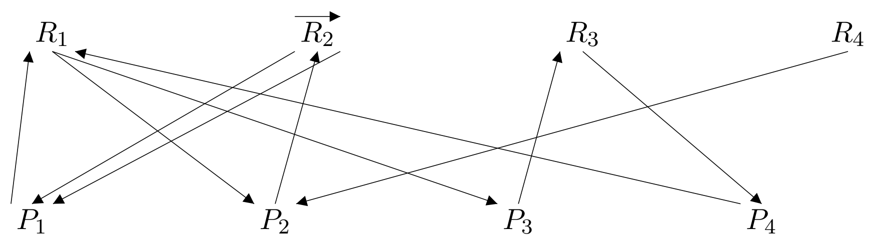
- R_i: request vector for process P_i

- A_i: current allocation vector for process P_i
- U: unallocated (available) resource vector

Given the scenarios below, draw the corresponding resource allocation graph. Indicate if the system is deadlocked and justify your answer (one sentence).

a. **(5 marks)** $U = (0, 1, 0, 0)$
$R_1 = (1, 0, 0, 0)$, $R_2 = (0, 1, 0, 0)$, $R_3 = (0, 0, 1, 0)$, $R_4 = (1, 0, 0, 0)$
$A_1 = (0, 2, 0, 0)$, $A_2 = (1, 0, 0, 1)$, $A_3 = (1, 0, 0, 0)$, $A_4 = (0, 0, 1, 0)$.

————————begin solution————————



The system is NOT deadlocked.

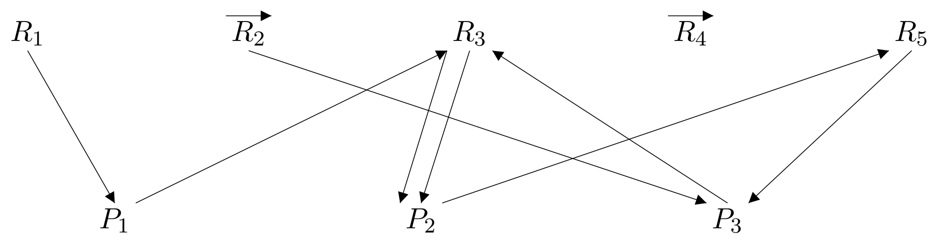
P_2 can be satisfied using the unallocated ressource R_2,
so that it will free at some point its ressoures R_1, R_2, R_4,
so that P_1 can be satisfied in turn and release ressources $R_1, 2 * R_2$,
so that P_4 can be satisfied in turn and release ressource R_1, R_3,
so that finally P_3 can be satisfied.

If U is modified to $(0, 0, 0, 0)$, then indeed the system would be deadlocked.

————————end solution————————

b. **(5 marks)** $U = (0, 1, 0, 1, 0)$
$R_1 = (0, 0, 1, 0, 0)$, $R_2 = (0, 0, 0, 0, 1)$, $R_3 = (0, 0, 1, 0, 0)$
$A_1 = (1, 0, 0, 0, 0)$, $A_2 = (0, 0, 2, 0, 0)$, $A_3 = (0, 1, 1, 0, 1)$.

————————begin solution————————



The system is deadlocked.

All the resources R_3 and R_5 are hold by P_2 or P_3, and
P_3 is waiting for R_5 which is hold by P_2,
itself waiting for R_3 which is hold by P_3.
P_1 can not make any progress to release any resources that would help either P_2 or P_3 because it is also waiting for R_3. Even if P_1 was able to make progress (which it is not) it is not allocated any resources whose availability would help P_2 or P_3.

————————end solution————————