CS1356 Introduction to Information Engineering

Homework 5

Due: Dec 16, 2009 in class

Remember to write your name and student ID

1. We call a DVD is played smoothly if it can be played 30 frames per second *uniformly*. Suppose you want to play two DVDs simultaneously on a single core computer, which takes 10 ~~microseconds~~ms to display a frame, and takes 1 ~~microsecond~~ms for context switch. What is the longest length of a time slice to make both DVDs played smoothly? Assuming no other programs are executed at the same time. And what is the shortest length of a time slice to do so? Justify your answers. (**20%**)

   Ans: There must be at least 60 time slices per second to make 2 DVDs played smoothly. Otherwise, the frames cannot be played uniformly.
   So the longest length of a time slice is (1000ms – 1ms*60)/60 = 15.67ms.

   On the other hand, the minimum length of time to play 2 DVD per second is 10ms*30 frames*2 DVD = 600ms. Thus, the maximum length of time for context switch is 1000ms – 600ms=400ms, so there are at most 400ms/1ms=400 context switches. According to that, the shortest length of a time slice is (1000ms – 1ms*400)/400=1.5ms.

   *** In the original settings, the answers will be 16.67ms and 0.6004ns, which is unreasonable. But the main idea is the same: the length of a time slice cannot be too long and cannot be too short.

2. A *barrier* for a group of threads/processes in the source code means any thread/process must stop at this point and cannot proceed until all other threads/processes reach this barrier. The following code uses a global variable `count`, whose initial value is 8, to implement a barrier for 8 threads/processes. (**20%**)

   (1) Give an example to show this code could fail in a single CPU multitasking environment.
   (2) How to fix this code to make it work? Justify your answers.

```
count = count - 1;
while (count > 0);
// barrier point
```

Ans: (1) In the instruction level, the statement "`count=count-1;`" will be translated into at least 3 instructions.

```
1. Read variable count from memory to a register, say R1
2. Perform R3 = R1-R2 (assume the value in R2 is 1)
3. Store the value in R1 to the memory cell of count
```

A scenario that fails the statements is when one process, say process A, had executed instruction 1, and then the OS context switched to another process, say process B; and process B finishes instruction 1, 2, 3, and then the OS context switched back to process A.

To make it clear, let us assume that count = 2 in the beginning of this scenario. And the execution history like the follows.

Executed process	Executed instruction	Value of `count` in memory
A	Instruction 1	2
B	Instruction 1	2 (R1=2 now)
B	Instruction 2	2 (R3 = 1)
B	Instruction 3	1 (R3 is written back)
A	Instruction 2	1 (R3 = 1)
A	Instruction 3	1 (R3 is written back)

(2) You can use either test_set() function, or use the disable_interrupt() and enable_interrupt() functions, to enclose the "`count=count-1;`" statement. But if you use test_set(), you need to use while-loop to keep trying.

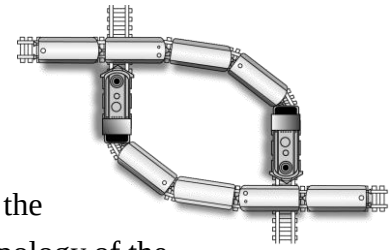
```
while (1){
   if (test_set(flag)) {
      count = count - 1;
      break;
   }
}
flag = 0;
while (count > 0);
```

```
disable_interrupt();
count = count - 1;
enable_interrupt();
while (count > 0);
```

** There is a bug in the program which is usually overlooked. That is, most compilers will use the value in registers to execute the statements "`count=count-1; while (count> 0);`", which will make the changes in memory useless. To make compiled code to read variable from memory every time, the modifier "volatile" should be used in the declaration of variable "count".

```
volatile int count = 0;
```

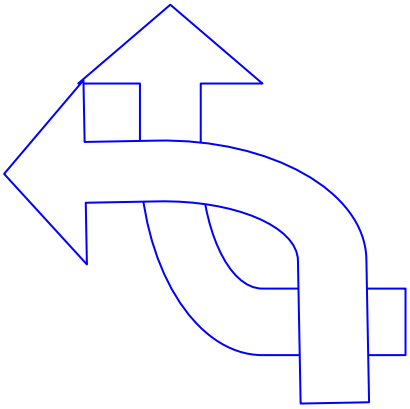
3. Textbook uses an example to illustrate the deadlock situation, as shown in the figure. The solutions to resolve deadlock are attacking one of the following conditions.



(1) Competition for non-sharable resources
(2) Resources requested on a partial basis
(3) Allocated resources cannot be forcibly retrieved.
(4) Circular wait.

For each condition, can you think a solution to attack it so that the deadlock is resolved? Your solutions CANNOT change the topology of the railroads, but you can add new facilities to it, or design new rules for train operations. (**40%**)

Ans: Here are just some examples for each case.



(1) Add overpasses or underpasses to the crossed railroad.
(2) Use a traffic light to indicate if there is any train passing the crossed section. If so, other train cannot pass
(3) Let one of the train back up when a deadlock happens
(4) Do not allow both trains go from opposite directions. For instance, you only allow both trains go from bottom to top, as shown in the figure.

4. CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*) is a system designed to distinguish computer and human. A common type of CAPTCHA requires user to type letters or digits from a distorted image, as the following one. (**20%**)

(1) Nowadays, this technique is often used with the authentication process. What kind of security attack can this mechanism prevent?
(2) Explain how this technique can be used to attack (email) spam filters?

Ans: (1) This can be used to prevent robot programs to attack servers, such as dictionary attack or sending spam emails.
(2) Since most spam filter will check the content of emails, if you use the distorted image in the email, the spam filter will not be able to know what the content is.