

Quiz 5

Course: cs241 - System Programming, CS Department

Date: March 5, 2006

Netid:

Name:

UIN:

Note: Completion of quiz is an individual effort. The quiz takes 10 minutes. The student gets additional 5 points for taking the quiz. Each question has ONLY ONE ANSWER!!!

- (1 Point) Given a system where all shared resource types are numbered, if a process is ordering a resource of *type N*, it should not be holding a resource of type greater than or equal to *N*. This is a method for deadlock
 - Prevention**
 - Detection
 - Recovery
- (1 Point) A deadlock detection algorithm is run
 - Every time a resource is allocated
 - Every time a resource is released
 - As frequently as is decided by the system administrator**
- (1 Point) Let us consider two processes *A* and *B* and three resource types *R*, *S*, and *T*. Let us consider that process *A* requests resource types *R*, *S*, and *T*, and process *B* requests two of the resource types *R* and *S*. If *A* gets assigned resource types *R* and *T*, and *B* gets assigned resource type *S*, then
 - A* and *B* will be in deadlock situation**
 - There is no deadlock situation between *A* and *B*
- (2 Points) To measure the running time of a function `function_to_time` by using POSIX:TMR clocks we need to run the following piece of code (error checking is omitted):

```
long timedif;
struct timespec ovalue; nvalue;

clock_gettime(CLOCK_REALTIME, &ovalue);
function_to_time();
clock_gettime(CLOCK_REALTIME, &nvalue);
```

To compute the running time in microseconds the following formula is correct:

- `timedif = MILLION*(nvalue.tv_sec - ovalue.tv_sec) + (nvalue.tv_usec - ovalue.tv_usec)/1000;`
- `timedif = MILLION*(ovalue.tv_sec-nvalue.tv_sec) + (ovalue.tv_nsec - nvalue.tv_nsec)`
- `timedif= MILLION*(nvalue.tv_sec-ovalue.tv_sec) + (nvalue.tv_nsec - ovalue.tv_nsec)/1000`**
- `timedif = MILLION*(ovalue.tv_sec - nvalue.tv_sec) + (ovalue.tv_usec - nvalue.tv_usec)`

5. (1 Point) Interval timers are useful in programming
 - a. Queue structures
 - b. Process schedulers**
 - c. Convey effects

6. (1 Point) To get into a deadlock situation, we need to have
 - a. Mutual exclusion AND hold and wait situation AND no preemption AND circular wait situation**
 - b. Mutual exclusion AND preemption AND spooling everything AND ordering of resources numerically
 - c. No preemption AND hold and wait situation AND no preemption AND ordering of resources numerically

7. (1 Point) What is the difference when measuring run time of a **function_to_time** using POSIX:XSI timer functions versus POSIX:TMR timer functions?
 - a. XSI timer is more precise than TMR timer
 - b. TMR timer is more precise than XSI timer**
 - c. TMR timer can use virtual time where XSI timer is allowed to use only clock real-time.

8. (1 Point) What does the following code do (error checking omitted) when we pass **sec** parameter with value 2.5?

```

static int functionE(double sec) {
timer_t timerid;
struct itimerspec tpend;

timer_create(CLOCK_REALTIME, NULL, &timerid);
tpend.it_interval.tv_sec = (long) sec;
tpend.it_interval.tv_nsec = (sec -
tpend.it_interval.tv_sec)*BILLION;
tpend.it_value = tpend.it_interval;
return timer_settime(timerid,0,&tpend, NULL);
}

```

- a. Set the timer to the current time
- b. Set the timer to the interval equal to the value stored in **CLOCK_REALTIME**
- c. Set the timer to the interval equal to the value of 2.5**
- d. Set the timer to the interval equal to the value stored in **timerid**;

9. (1 Point) What does the following code segment do (error handling is omitted) ?

```
struct sigaction newact;  
newact.sa_handler = SIG_DFL;  
newact.sa_flags = 0;  
sigemptyset(&newact.sa_mask) || sigaction(SIGINT,  
&newact, NULL);
```

- a. set the signal handler to print {*} in the signal handler;
- b. assign signal **SIGINT** to the signal mask in the signal handler;
- c. set the action of SIGINT to be the default signal handler;**