FALL 2014 CSCI-4120/6140

Sample midterm exam questions

========================================================

WHAT IS THE EXACT OUTPUT OF THE FOLLOWING CODE?

FOR ANY FILES WRITTEN TO, WHAT ARE THEIR EXACT CONTENTS?

========================================================

/\* midterm-ex00.c \*/

#include <stdio.h>

#include <sys/types.h>

#include <stdlib.h>

int main()

{

int x = 150;

printf( "PARENT: x is %d\n", x );

printf( "PARENT: forking...\n" );

pid\_t pid = fork();

printf( "PARENT: forked...\n" );

if ( pid == 0 )

{

printf( "CHILD: happy birthday\n" );

x \*= 2;

printf( "CHILD: %d\n", x );

}

else

{

wait( NULL );

printf( "PARENT: child completed\n" );

x \*= 2;

printf( "PARENT: %d\n", x );

}

return 0;

}

PARENT: x is 150

PARENT: forking...

PARENT: forked...

CHILD: happy birthday

CHILD: 300

PARENT: forked...

PARENT: child completed

PARENT: 300

**Correct:**

**PARENT: x is 150**

**PARENT: forking...**

**PARENT: forked...**

**PARENT: forked...**

**CHILD: happy birthday**

**CHILD: 300**

**PARENT: child completed**

**PARENT: 300**

====================================================

/\* midterm-ex01.c \*/

#include <stdio.h>

#include <unistd.h>

int main()

{

int rc;

printf( "ONE\n" );

rc = fork();

printf( "TWO\n" );

if ( rc == 0 ) { printf( "THREE\n" ); }

if ( rc > 0 ) { printf( "FOUR\n" ); }

return 0;

}

ONE

TWO

FOUR

TWO

THREE

====================================================

/\* midterm-ex02.c \*/

#include <stdio.h>

#include <unistd.h>

int main()

{

printf( "ONE\n" );

fprintf( stderr, "ERROR: ONE\n" );

int rc = close( 1 );

printf( "TWO\n" );

fprintf( stderr, "ERROR: TWO\n" );

rc = dup2( 2, 1 );

printf( "THREE\n" );

fprintf( stderr, "ERROR: THREE\n" );

rc = close( 2 );

printf( "FOUR\n" );

fprintf( stderr, "ERROR: FOUR\n" );

return 0;

}

ERROR: ONE

ERROR: TWO

ONE

TWO

THREE

ERROR: THREE

**Correct:**

**ONE**

**ERROR: ONE  
ERROR: TWO**

**THREE**

**ERROR: THREE**

**FOUR**

====================================================

/\* midterm-ex03.c \*/

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

int main()

{

int fd;

close( 2 );

#if 0

close( 1 ); /\* <== add this line later (part b) \*/

#endif

printf( "HI\n" );

fflush( stdout );

fd = open( "newfile.txt", O\_WRONLY | O\_CREAT | O\_TRUNC, 0600 );

printf( "%d\n", fd );

printf( "WHAT?\n" );

fprintf( stderr, "ERROR\n" );

return 0;

}

HI

2

WHAT?

(To the newfile.txt) ERROR

(Part b)

(To the new file.txt)

1

WHAT?

====================================================

/\* midterm-ex04.c \*/

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/wait.h>

int main()

{

int fd;

close( 2 );

#if 0

close( 1 ); /\* <== add this line later (part b) \*/

#endif

printf( "HI\n" );

fflush( stdout );

fd = open( "newfile.txt", O\_WRONLY | O\_CREAT | O\_TRUNC, 0600 );

printf( "%d\n", fd );

printf( "WHAT?\n" );

fprintf( stderr, "ERROR\n" );

fflush( stdout );

int rc = fork();

if ( rc == 0 )

{

printf( "AGAIN?\n" );

fprintf( stderr, "ERROR ERROR\n" );

}

else

{

wait( NULL );

}

printf( "BYE\n" );

fprintf( stderr, "HELLO\n" );

return 0;

}

HI

ERROR

2

WHAT?

ERROR ERROR

AGAIN?

HELLO

BYE

**Correct:**

**HI**

**2**

**WHAT?**

**AGAIN?**

**BYE**

**BYE**

====================================================

/\* midterm-ex05.c \*/

#include <stdio.h>

#include <unistd.h>

int main()

{

int rc;

int p[2];

rc = pipe( p );

printf( "%d %d %d\n", getpid(), p[0], p[1] );

rc = fork();

if ( rc == 0 )

{

rc = write( p[1], "ABCDEFGHIJKLMNOPQRSTUVWXYZ", 26 );

}

if ( rc > 0 )

{

char buffer[40];

rc = read( p[0], buffer, 8 );

buffer[rc] = '\0';

printf( "%d %s\n", getpid(), buffer );

}

printf( "BYE\n" );

return 0;

}

3 4

ABCDEFGH

BYE

**Correct:**

**4470 3 4**

**4471 ABCDEFGH**

**BYE**

**4470 IJKLMNOP**

**BYE**

========================================================

1. Given the table of process burst times, arrival times,

and priorities below, calculate the turnaround time,

total wait time, and number of preemptions for each

process using the following scheduling algorithms:

a. FCFS

b. SJF

c. SRT

d. Priority scheduling with SJF (note that the

lower the priority value, the higher priority

the process has)

e. HRRN

f. RR with a time slice of 3ms

PROCESS BURST TIME ARRIVAL TIME PRIORITY

----------------------------------------------

P1 7ms 0ms 2

P2 5ms 0ms 3

P3 5ms 1ms 1

P4 6ms 4ms 2

FCFS

turn around time total wait time number of preemptions

P1 7 0 0

P2 12 7 0

P3 16 11 0

P4 19 13 0

SJF

turn around time total wait time number of preemptions

P1 23 16 0

P2 5 0 0

P3 9 4 0

P4 12 6 0

SRT

turn around time total wait time number of preemptions

P1 23 16 0

P2 5 0 0

P3 9 4 0

P4 12 6 0

Priority SJF

turn around time total wait time number of preemptions

P1 19 12 0

P2 23 18 0

P3 5 0 0

P4 8 2 0

HRRN

turn around time total wait time number of preemptions

P1 7 0 0

P2 12 7 0

P3 16 11 0

P4 19 13 0

RR

turn around time total wait time number of preemptions

P1 20 13 2

P2 17 12 1

P3 18 13 1

P4 19 13 1

2. Write pseudocode for the scheduling algorithms

above; for each, determine what the computational

complexity of adding a process to the ready queue,

of selecting the next process, and of removing

a process from the queue.

3. For each scheduling algorithm above, determine (and

describe how) whether CPU-bound or I/O-bound processes

are favored.

4. For each scheduling algorithm above, describe how

starvation can occur (or describe why it never will

occur).

5. Given actual burst times for process P1 shown below

and an initial guess of 6ms, calculate the estimated

next burst time for P1 by using exponential averaging

with alpha (a) set to 1/2; recalculate with alpha set

to 1/4 and 3/4.

Measured burst times for P1: 12ms, 8ms, 8ms, 9ms, 4ms

6. Describe what happens when a child process terminates;

also, what happens when a process with child processes

terminates?

7. In a shell, how might the pipe functionality actually

be implemented? Describe in detail.

8. Why might fork() fail? Why does a "fork-bomb" cause

a system to potentially crash?