PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
MISSISSAUGA
APRIL 2008 EXAMINATIONS

CSC 209H5S
Instructor — Michelle Craig
Duration — 3 hours

Examination Aids: one 8.5 x 11 inch double-sided page. No electronic aids.

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Last (Family) Name(s): _____

First (Given) Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below *carefully*.)

This final examination consists of 8 questions on 17 pages (including this one). *When you receive the signal to start, please make sure that your copy of the examination is complete.*

You may be charged with an academic offence for possessing the following items during the writing of an exam unless otherwise specified: any unauthorized aids, including but not limited to calculators, cell phones, pagers, wristwatch calculators, personal digital assistants (PDAs), iPODS, MP3 players, or any other device. If any of these items are in your possession in the area of your desk, please turn them off and put them with your belongings at the front of the room before the examination begins. A penalty MAY BE imposed if any of these items are kept with you during the writing of your exam.

Please note, students are NOT allowed to petition to RE-WRITE a final examination.

You may assume that all necessary header files have been included.

MARKING GUIDE

1: _____/ 8

2: _____/ 8

3: _____/ 6

4: _____/ 8

5: _____/10

6: _____/15

7: _____/10

8: _____/15

TOTAL: _____/80

Question 1. [8 MARKS]

Write a bash script that requires at least three arguments. The first argument is a pattern string `p`, the second is an integer `n` and the remainder are directories. The script must print to standard output the names of regular files in those directories that contain the string `p` and are fewer than `n` blocks. It should print a message and exit if called with fewer than 3 arguments. Note that calling the command `du` prints the size of the file in blocks and the filename separated by whitespace.

Question 2. [8 MARKS]

Write a C program that takes at most two arguments. The first argument (which is required) is a string `p`. The second argument is a path to a directory `d`. It is optional. The program should list all the files in directory `d` whose names contain the string `p`. If the program is called with an invalid number of arguments, it should print a usage message to standard error and exit. If the optional directory is not provided, it should use the current working directory.

Question 3. [6 MARKS]

Consider the following struct and type definition.

```
struct studentNode{
    char name[20];
    char university[30];
    struct studentNode *next;
};

typedef struct studentNode NN;
```

Complete the function `find_school` so that it creates and returns a string holding the university for the first student with the given name from the unsorted linked list. The function should allocate only enough space to hold the name of the school. The function should return `NULL` if there is no such student in the list.

```
/*
 * Create a new string set equal to the school name of the
 * first student with name st_name in list st_list.
 * If there is no such student, return NULL.
 * Do not change the list.
 *
 */
char * find_school(NN * st_list, char * st_name) {
```

Question 4. [8 MARKS]

Part (a) [5 MARKS]

In the next question, there will be a lot of pipes to close. Write a wrapper function called `Close` that has the string `msg` as one of its formal parameters. Within your function, call the `close` system call and perform error checking. If you encounter an error, print a message to standard error that includes the string "Close:" appended with the string from `msg` along with the standard message associated with `errno`.

Part (b) [1 MARK]

Explain why it is important to close the writing end of a pipe in a process which is going to use the pipe for reading.

Part (c) [1 MARK]

In assignment three, some solutions waited for the map processes to complete before reading from any of the pipes in the reduce processes. This appeared to work on some test cases but failed on others. Explain why.

Question 5. [10 MARKS]

The C program below is intended to create 5 child processes, send each of them the message "hello" using a pipe and read an integer returned from each child. Fill in the missing parts of the program so that it works correctly according to the comments in the code. Assume that the necessary includes and headerfiles are already in place. Some of the code is formatted oddly to allow it to fit on the page. There should be lots of room for your code.

```
#define NUMKIDS 5
int main() {
  int pid, i;
  // declare the pipes needed for 2-way communication
  int tochild[NUMKIDS][2];
  int tomaster[NUMKIDS][2];

  for (i=0; i<NUMKIDS; i++) {
```

```
//create the pipes needed to communicate with this next child
```

```
    if ((pid = fork())<0) { perror("fork"); exit(1); }
    if (pid == 0) {
```

```
   // close the pipes not needed here            (use the Close function from Question 4)
```

```
       // now call the do_child method filling in the parameters to match
       // match the signature  do_child(int child_num,int infd,int outfd)
       do_child(                 ,                          ,                        );
```

```
       exit(0);                          // LINE A MARKED FOR LATER QUESTION
    } // bracket matches if (pid == 0)
```

 CONT'D ON PG 7

```
    else {
```

```
        //close pipes that aren't needed here
```

```
   } // matches else
  } // matches for
  // all five children created with full duplex communication
```

```
  // send "hello" to each child
  // you may have more pipes to close first depending on your earlier decisions
```

```
  int total=0;
```

```
  // read an integer from each child and add them to total
```

```
  printf("total from my children\n",total);
  return 0;                        //LINE B MARKED FOR LATER QUESTION
} // matches main
```

Part (a) [1 MARK]

In the previous program, there is a line marked `// LINE A MARKED FOR LATER QUESTION` . What would happen if this line was removed from the program?

Part (b) [1 MARK]

In the previous question, there is a line marked `// LINE B MARKED FOR LATER QUESTION` . What would happen if this line was removed from the program?

Question 6. [15 MARKS]

For each code fragment below, indicate whether it WORKS GREAT, WORKS BUT HAS ISSUES, or NEVER WORKS. Then, explain the reasoning behind your answer. Answers without explanations will receive no marks. For the category WORKS BUT HAS ISSUES, it might be the case that the code works sometimes and not others, or it might be that the code runs but doesn't do what the user is expecting as described in the comments. ' is a single quote, and ‘ is a back quote.

Part (a) [1 MARK]

```
//reset behaviour for SIGKILL to be ignored
if (sigaction(SIGKILL, SIG_IGN, NULL) != 0) {
  perror("sigaction");
  exit(1);
}
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Part (b) [2 MARKS]

```
# argument to the script is a filename in the current directory
# print this filename and the owner of the file
# reminder ls -l format is:
# -rw-r--r--  1 joeowner  somegrp  740 28 Nov 04:44 somefile.txt
echo $1
set ‘ls -l $1‘
echo $3
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Part (c) [2 MARKS]

```
# argument to the script is a filename in the current directory
# print this filename and the owner of the file
# reminder ls -l format is:
# -rw-r--r--  1 joeowner  somegrp  740 28 Nov 04:44 somefile.txt
set ‘ls -l $1‘
echo $1 $3
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Part (d) [2 MARKS]

```
file=analysis.txt
words="wc -w $file"
echo There were $words words in $file
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Part (e) [2 MARKS]

```
char s[25] = "2001: A Space Odyssey";
char * n = malloc(strlen(s) + 1);
char * p = s;
int c = 0;
while ((*n = *p) != '\0') {
  p++; n++; c++;
}
n -= c;
printf("n now points to %s\n",n);
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Part (f) [2 MARKS]

```
struct rec {
  char name[20]; int age;
};

int main() {
  struct rec *a, *p;
  a = malloc(10*sizeof(struct rec));

  ... assign values to the 10 struct rec names ...

  // print out all the names
  for (p=a; p < a+10; p++) {
     printf("%s\n",p->name);
  }
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Part (g) [1 MARK]

```
char * s = "hotdogs";
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Part (h) [1 MARK]

```
char p[7] = "mustard";
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Part (i) [1 MARK]

```
#print date in format Mon dd, yyyy
#reminder format of date output is: Tue Mar 18 10:05:47 EDT 2008
set ‘date‘
shift; echo $1
shift; echo $1,
shift; shift; shift;
echo $1
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Part (j) [1 MARK]

```
// return pointer to the struct containing the longer run
struct run * longer(struct run r1, struct run r2) {
  if (r1.miles > r2.miles)
    return &r1;
  else
    return &r2;
}
```

WORKS GREAT ☐ WORKS BUT HAS ISSUES ☐ NEVER WORKS ☐

Explanation:

Question 7. [10 MARKS]

Write a C program that takes an integer argument `s` representing a number of seconds and a string representing an existing filename. The program repeatedly generates a random integer from 0 to 99 and reads an integer starting from that byte offset in the file. The program runs for `s` seconds (of real time not virtual time) and then stops and reports how many reads were completed. You may assume that the file is at least 104 bytes long.

Hint: We have provided an abridged manual page for `setitimer` at the back of the exam.

Question 8. [15 MARKS]

For each code fragment or statement below, a subsection is indicated with a box. Explain what the boxed code does and why it is important. It is insufficient to simply say that a variable is assigned a value or a function is called. Your explanation should be clear about why the code is included in the program.

Part (a) [1 MARK]

```
fd_set      rset, allset;

FD_ZERO(&allset);
FD_SET(listenfd, &allset);

for ( ; ; ) {
  rset = allset;
  nready = Select(maxfd+1,  &rset, NULL, NULL, NULL);
```

Part (b) [2 MARKS]

```
fd_set      rset, allset;
FD_ZERO(&allset);
FD_SET(listenfd, &allset);
for ( ; ; ) {

  rset = allset;
  nready = Select(maxfd+1,  &rset, NULL, NULL, NULL);
```

Part (c) [1 MARK]

```
    for ( ; ; ) {
        rset = allset;
        nready = Select(maxfd+1, &rset, NULL, NULL, NULL);

        if (FD_ISSET(listenfd, &rset)) {        /* new client connection */
```

Part (d) [2 MARKS]

```
struct sigaction newact;
sigemptyset(&newact.sa_mask);
newact.sa_handler = sig_quit;
newact.sa_flags = 0;
if(sigaction(SIGQUIT, &newact, NULL) == -1)
     perror("Could not install handler for SIGQUIT");
```

Part (e) [2 MARKS]

```
struct sigaction newact;
sigemptyset(&newact.sa_mask);
newact.sa_handler = sig_quit;
newact.sa_flags = 0;
if(sigaction(SIGQUIT, &newact, NULL) == -1)
     perror("Could not install handler for SIGQUIT");
```

Part (f) [1 MARK]

```
struct sockaddr_in self;
self.sin_port = htons(PORT);
```

Part (g) [2 MARKS]

```
while((ch = getopt(argc, argv, "bd:f")) != -1)
```

Part (h) [2 MARKS]

```
int status;
wait(&status);
if (WIFEXITED(status)) {
  printf("exit status %d\n", WEXITSTATUS(status));
```

Part (i) [2 MARKS]

```
void printArray(int a[12], int size) {
  int i;
  for (i=0;i<size;i++) {
    printf("A[%d] is %d\n",i,a[i]);
  }
}
```

Use this page for rough work or answers that did not fit in the allocated space. Clearly indicate where something is to be marked.

Total Marks = 80

GETITIMER(2) Linux Programmers Manual GETITIMER(2)

NAME

getitimer, setitimer - get or set value of an interval timer

SYNOPSIS

```
#include <sys/time.h>

int getitimer(int which, struct itimerval *value);
int setitimer(int which, const struct itimerval *value,
              struct itimerval *ovalue);
```

DESCRIPTION

The system provides each process with three interval timers, each decrementing in a distinct time domain. When any timer expires, a signal is sent to the process, and the timer (potentially) restarts.

ITIMER_REAL decrements in real time, and delivers SIGALRM upon expiration.

ITIMER_VIRTUAL decrements only when the process is executing, and delivers SIGVTALRM upon expiration.

ITIMER_PROF decrements both when the process executes and when the system is executing on behalf of the process. Coupled with ITIMER_VIRTUAL, this timer is usually used to profile the time spent by the application in user and kernel space. SIGPROF is delivered upon expiration.

Timer values are defined by the following structures:

```
struct itimerval {
    struct timeval it_interval; /* next value */
    struct timeval it_value;    /* current value */
};
struct timeval {
    long tv_sec;                /* seconds */
    long tv_usec;               /* microseconds */
};
```

The function getitimer() fills the structure indicated by value with the current setting for the timer indicated by which (one of ITIMER_REAL, ITIMER_VIRTUAL, or ITIMER_PROF). The element it_value is set to the amount of time remaining on the timer, or zero if the timer is disabled. Similarly, it_interval is set to the reset value. The function setitimer() sets the indicated timer to the value in value. If ovalue is non-zero, the old value of the timer is stored there.

Timers decrement from it_value to zero, generate a signal, and reset to it_interval. A timer which is set to zero (it_value is zero or the timer expires and it_interval is zero) stops.

Both tv_sec and tv_usec are significant in determining the duration of a timer.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

ERRORS

EFAULT value or ovalue are not valid pointers.

EINVAL which is not one of ITIMER_REAL, ITIMER_VIRTUAL, or ITIMER_PROF.

NOTES

A child created via fork(2) does not inherit its parents interval timers. Interval timers are preserved across an execve(2).