PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

AUGUST 2012 EXAMINATIONS

CSC 209H1Y
Instructor: Daniel Zingaro

Duration — three hours

Examination Aids: one two-sided 8.5x11 sheet.

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

Last (Family) Name(s): _____

First (Given) Name(s): _____

---

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

This examination consists of 9 questions, on 18 pages (including this page).

Instructions:

- Check to make sure that you have all 18 pages.
- Read the entire exam before you start.
- Not all questions are of equal value, so budget your time accordingly.
- Do not add `include` lines. Add error-checking only if indicated by the question.
- If you use any space for rough work, indicate clearly what you want marked.

MARKING GUIDE

# 1:	_____/11
# 2:	_____/12
# 3:	_____/12
# 4:	_____/ 9
# 5:	_____/ 7
# 6:	_____/ 7
# 7:	_____/ 8
# 8:	_____/13
# 9:	_____/ 1
TOTAL:	_____/80

Good Luck!

CONT'D...

Question 1. [11 MARKS]

Part (a) [2 MARKS]

The following two shell commands do different things. Assume that `foo` is an executable program with execute permissions.

```
foo >out 2>&1
foo 2>&1 >out
```

Briefly describe the result of each command and why they are different.

Part (b) [2 MARKS]

The shell code below tries to output the value of `quad`, but fails.

```
greywolf:~$ IFS=x # don't change this line
greywolf:~$ quad=axbxc
greywolf:~$ echo $quad
a b c
```

Add the proper quoting so that `axbxc` is properly output.

Part (c) [2 MARKS]

State two differences between `read` and `fgets`.

Part (d) [1 MARK]

Give one difference between `SIGTSTP` and `SIGSTOP`.

Part (e) [1 MARK]

How many file descriptors are closed when I call `close` on a socket file descriptor?

Part (f) [3 MARKS]

Assume that I try to send a message using UDP. Based on the fact that message delivery is unreliable in UDP, describe several possibilities for what could happen to my message.

Question 2. [12 MARKS]

Here is part of the man page for a C function called `strsep`. (You used this in lab 4 when learning how to tokenize strings.)

```
SYNOPSIS
#include <string.h>

char *strsep(char **stringp, const char *delim);

DESCRIPTION
If  *stringp is NULL, the strsep() function returns NULL and does
nothing else.  Otherwise, this function finds the first token in the string
*stringp,  where  tokens  are delimited by symbols in the string delim.
This token is terminated by overwriting the delimiter with a null  byte
('\0')  and  *stringp  is  updated to point past the token.  In case no
delimiter was found, the  token  is  taken  to  be  the  entire  string
*stringp, and *stringp is made NULL.

RETURN VALUE
The  strsep()  function  returns  a  pointer  to the token, that is, it
returns the original value of *stringp.
```

Part (a) [6 MARKS]

Write an implementation of `strsep`. You may use `strstr` or `strchr`, but do not use any other string functions.

```
char *mystrsep(char **stringp, const char *delim) {
```

Part (b) [6 MARKS]

Write the following function that parses a line of text (`line`) into its tokens and stores the tokens in `argv`. Assume that `argv` has enough room to point to the tokens and that `line` is not a string constant. Tokens are separated by spaces or enclosed in single quotes. For example, the following `line` string has four tokens (not five!):

```
'Armos Knights' Moldorm Helmasaur Mothula
```

You must use `strsep` to tokenize `line`. You may assume that `line` does not have any leading or trailing spaces and that there is exactly one space between each pair of tokens. `tokenize` should return the number of tokens found.

```
int tokenize(const char *line, char **argv) {
```

Question 3. [12 MARKS]

Consider the following `typedef`, which creates a new type called `Student`.

```
typedef struct student {
  char *first, *last;
  struct student *next;
} Student;
```

Part (a) [3 MARKS]

Assume that `char` pointers are four bytes and `Student` pointers are also four bytes. What is the total amount of memory occupied by the following declarations? Assume that all `malloc` calls succeed. Please show your work.

```
Student *a = malloc(sizeof (Student));
a->first = malloc(10);
a->last = NULL;
Student *b = a;
b->last = malloc(15);
```

Part (b) [4 MARKS]

`sl` is a linked list (i.e. it ends with `NULL`) of `Students`. Write a function `freeList` that frees all memory used by the list.

```
void freeList (Student *sl) {
```

Part (c) [5 MARKS]

Imagine that a list of students could be in one of two forms: a regular linked list like above, or a circular linked list like in Solitaire. Write a function `isCircular` that returns a true value if the list is circular and a false value if the list is not circular.

```
int isCircular(Student *sl) {
```

Question 4. [9 MARKS]

The code on this page is used by all parts of this question. Error-checking is not included; assume that no errors occur.

```
int main(void) {
  int i, infd, outfd;
  int fds[2];
  char buf[6];
  for (i = 0; i < 2; i++) {
    pipe (fds);
    if (fork() > 0) {
      outfd = dup(fds[1]);
      // A: close (fds[0]);
      // B: close (fds[1]);
      if (i == 0)
        write (outfd, "hello", 5);
      else {
        read (infd, buf, 5);
        write (outfd, buf, 5);
      }
      exit(0);
    }
    // C: close (infd);
    infd = dup(fds[0]);
    // D: close (fds[0]);
    // E: close (fds[1]);
  }
  read (infd, buf, 5);
  write (STDOUT_FILENO, buf, 5); // *
  return 0;
}
```

Part (a) [2 MARKS]

How many times is `fork` executed?

Part (b) [2 MARKS]

Consider the line at the bottom (the line with the `// *`). Write alternate code (one or more lines) that uses `printf` instead of `write`.

Please answer **yes** or **no** to each part on this page. These correspond to the five A-E comment lines in the code. Each part is to be considered separately. Answer **yes** if both the `close` will be successful and the program will still print `hello` on standard output. Otherwise, answer **no**.

Part (c) [1 MARK]

Is it correct to uncomment the `// A` line? **yes** **no**

Part (d) [1 MARK]

Is it correct to uncomment the `// B` line? **yes** **no**

Part (e) [1 MARK]

Is it correct to uncomment the `// C` line? **yes** **no**

Part (f) [1 MARK]

Is it correct to uncomment the `// D` line? **yes** **no**

Part (g) [1 MARK]

Is it correct to uncomment the `// E` line? **yes** **no**

Question 5. [7 MARKS]

Part (a) [1 MARK]

The `SIGCHLD` signal can inform you of processes that have stopped. (Circle one) **true** **false**

Part (b) [3 MARKS]

Consider the code below. Is it possible that the parent leaves one or more children in the zombie state? Please answer **yes** or **no** and briefly explain. Assume all syscalls succeed.

```
void reap(int code) {
  waitpid (-1, NULL, WNOHANG);
}

int main(void) {
  int i, n;
  struct sigaction newact;
  sigemptyset (&newact.sa_mask); newact.sa_flags = 0;
  newact.sa_handler = reap;
  if (sigaction(SIGCHLD, &newact, NULL) == -1)
    exit(1);

  for (i = 0; i < 10; i++) {
    n = fork();
    if (n == -1)
      exit (1);
    else if (n == 0)
      exit (0);
  }

  for(;;) ; //Infinite
}
```

Part (c) [3 MARKS]

Consider the code below. Again, is it possible that the parent leaves one or more children in the zombie state? Please answer **yes** or **no** and briefly explain. Assume all syscalls succeed.

```
void reap(int code) {
  waitpid (-1, NULL, WNOHANG);
}

int main(void) {
  int i, n;
  struct sigaction newact;
  sigemptyset (&newact.sa_mask); newact.sa_flags = 0;
  newact.sa_handler = reap;
  if (sigaction(SIGCHLD, &newact, NULL) == -1)
    exit(1);

  for (i = 0; i < 10; i++) {
    n = fork();
    if (n == -1)
      exit (1);
    else if (n == 0)
      exit (0);
    else {
      sigset_t set;
      sigfillset (&set);
      sigdelset (&set, SIGCHLD);
      sigsuspend (&set);
    }
  }

  for(;;) ; //Infinite
}
```

Question 6. [7 MARKS]

Consider text files where each line consists of a positive integer followed by a character. Here is one such file:

```
5c
2d
10e
1x
```

To **expand** such a file, we take each line and repeat the character the number of times specified by the integer. Expanding the above file gives us:

```
ccccc
dd
eeeeeeeeee
x
```

Write a shell script that supports the following synopsis:

```
expandrle filename [outfilename]
```

That is, the script takes the filename to expand, and an optional filename where the results of the expansion are to be placed. If `outfilename` is not provided, output should go into `filename` with an `.out` extension. Assume `filename` is readable and `outfilename` is writeable. If `outfilename` exists, it should be overwritten.

Question 7. [8 MARKS]

Part (a) [2 MARKS]

Describe the difference between level-triggered and edge-triggered notification.

Part (b) [3 MARKS]

Provide a brief scenario in which the `timeout` parameter to `select` could be useful. You are encouraged to use assignment 4 as your scenario (i.e. what would `timeout` allow you to do in assignment 4).

Part (c) [3 MARKS]

`fds` is an array of file descriptors all open for reading, and `n` is the number of FDs in `fds`. Write the required code so that the `select` at the bottom sets `set` to the subset of `fds` on which a `read` will not block. (What happens after the `select` is not important here.)

```
void doSomething (int fds[], int n) {
  // your code to make the select work









  nready = select(maxfd + 1, &set, NULL, NULL, NULL);
  ... don't write code here
}
```

Question 8. [13 MARKS]

Part (a) [5 MARKS]

Write a C program that takes a string as commandline argument and outputs on standard output the number of times that the string appears on standard input. For example, if I invoke the program as `countstring li` and provide the following on standard input:

```
Simplicity is prerequisite
for reliability.
```

then your program would output 3 on standard output. Assume that your program is called correctly with a nonempty string as the only argument.

```
int main(int argc, char *argv[]) {
```

Part (b) [5 MARKS]

In this part, you will write the body of a small server. Assume that all setup has been done for you and that you have a global socket file descriptor `listenfd` that is in the listening state.

Your server will be able to accept multiple concurrent clients, but you will not use `select` to do this. Instead, you will `fork` once for each connected client. When each client connects, your server should read everything on the client's socket until end-of-file. Once there is nothing left to read, you should send the client the number of times that the client sent you the string `test`.

You are to assume that `countstring` on the facing page has been compiled and is in the current directory, and you must exec this program for each child in order to read from and write to the socket. The following is an example of client interaction with your server (assuming you are listening on port 7006). The final line, `3`, is the response received by the client from the server.

```
redwolf:~$ /bin/nc redwolf.cdf.utoronto.ca 7006
test one two three
testing testing
<ctrl+d>
3
```

```
//accept(listenfd, (struct sockaddr *)&r, &socklen)
//make sure you set socklen = sizeof(r) before calling accept

int listenfd;

int main(void) {
  struct sockaddr_in r;
  socklen_t socklen;
  setup();
  //socket(), bind(), listen() done; listenfd is listening
  //Declare all variables and write code
```

Part (c) [3 MARKS]

In assignment 4, you used `select` to permit multiple connected clients. Presumably, you could have used `fork` as well, in the following way:

- Each time two opponents are matched and ready to battle, fork a new child
- That new child carries out the battle
- After the battle is complete, the child exits
- During this time, the parent is still listening for new clients and forking other child processes

Describe the particular difficulties in using this approach. What would have been trickier if you had used `fork` instead of `select`?

Question 9. [1 MARK]

Someday, when you reflect on CSC209, you'll hopefully remember something useful. Specifically, you're most likely to remember (circle one):

- Huh? CSC20what?
- The theme song
- The powermoves
- The segfaults
- Dan getting schooled even with his battle music

Thanks, folks. Onward!

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*