CSC209F Midterm (L5101)

Fall 1998

University of Toronto

Department of Computer Science

Date: November 2nd, 1998
Time: 6:10 pm
Duration: 50 minutes

Notes:

1. This is a closed book test, no aids are allowed.
2. Print your name, student number, and cdf username in the space provided below.
3. There are a total of 50 marks.
4. All shell questions assume `csh`.
5. This test is worth 20% of your final course mark.
6. This test comprises 7 pages. Do not detach pages, and answer questions in the blank spaces provided. If you need more space, answer on the back of the pages (clearly identify which question in this case).
7. **Read all questions before starting. Not all questions are worth the same number of marks, so budget your time accordingly. Answer questions in any order you desire.**

Name:	
Student Number:	
CDF Username:	

Marks:

Q1		Q5	
Q2		Q6	
Q3		Q7	
Q4			

Total

Q1: [10 marks]

1. How do you run a program in the background?
   Type '&' after the command, *e.g.* "`nedit &`", or type `^z` to suspend the process and then type "`bg`" to put the process into the background (*some people had ^z and bg reversed …*)

2. What does "`set noglob`" do when typed to the shell?
   Turns off filename expansion (*many thought this was related to global variables*)

3. (2 marks) Name 2 common UNIX shells.
   sh (Bourne), csh, tcsh, ksh

4. (2 marks) What is the difference between an absolute and relative pathname? Give an example of each.
   Absolute pathnames start with '/' and specifies a path starting from the root directory (*e.g.* /bin/csh). Relative pathnames specify a path starting in the current working directory (*e.g.* ../temp/myStuff.txt)

5. What is the command line to compile `myProg.c` so that the executable can be debugged using `gdb`?
   ```
   gcc -g myProg.c -o myProg
   ```
   where the -o option is optional (*many didn't get this … are you using the debugger for assignments?*)

6. What is the command to create a soft link "slink" to the file "/u/maclean/junk.txt"?
   ```
   ln -s /u/maclean/junk.txt slink
   ```

7. (2 marks) What is the value of argc and argv[2] in the C program invoked as
   "`myProg -13 fred < csc209.lst > csc209.marks`"?
   ```
   argc = 3
   argv[2] = "fred"
   ```
   (*Many got the argc value wrong … it's important to know that the program never "sees" I/O redirection …)*

Q2: [5 marks]

Briefly explain the I/O redirection occurring in each of the following commands:

`myCmd > someFile`
output (stdout) from "myCmd" goes into file "someFile" (created if necessary, or overwritten if it exists & "noclobber" is not set)

`myCmd < someFile`
input (stdin) for "myCmd" comes from file "someFile"

`myCmd >> someFile`
output (stdout) from "myCmd" is appended to file "someFile"

`myCmd >! someFile`
output (stdout) from "myCmd" overwrites "someFile", even if "noclobber" is set

`myCmd >& someFile`
output (stdout + stderr) from "myCmd" is written to "someFile"

In the first three parts I accepted "input" to mean stdin and output to mean "stdout", but the answers should have been more specific. In order to get the mark for the 4th one, the word "noclobber" had to be in your answer. In the last one many thought we were putting the process into the background.

Q3: [5 marks]

(3 marks) What are three possible outputs of:

```
int main(int argc, char *argv[])
{
  int pid ;
  if ((pid = fork()) == 0)
  {
    printf("I am = %d, Parent = %d\n", getpid(), getppid());
  }
  else
  {
    printf("Child = %d, I am = %d\n", pid, getpid());
  }
}
```

1) Assume child goes first (race condition):

```
I am = 10, Parent = 9
Child = 10, I am = 9
```

2) Parent goes first (race condition):

```
Child = 42, I am = 26
I am = 42, Parent = 26
```

3) Parent terminates before child prints (orphaned):

```
Child = 42, I am = 26
I am = 42, Parent = 1
```

4) Assume fork() is unsuccessful:

```
Child = -1, I am = 42
```

Many people only thought either the parent or the child got to printout, this is wrong. Some people had the parent's PID changing values, and some had the child with PID = 0 (this will never happen). Some thought that if `fork()` failed, the program would crash (not true).

(2 marks) How would you change the program to ensure a unique output?

Insert the lines

```
int status ;
wait(&status);
```

at the beginning of the `else` clause. The `wait()` call synchronizes the parent & child.

Q4: [5 marks]

Write a signal handler which catches interrupts typed at the keyboard (SIGINT is sent), and on the third interrupt prints "`3 strikes and you're out\n`", and causes the program to terminate. Show how your signal handler can be installed (a code fragment will suffice here). Recall that signal handlers are functions which have one `int` parameter and return `void`.

```
void handler(int signo)
{
  static int count = 0 ;

  if (++count == 3)
  {
    printf("3 strikes and you're out\n");
    exit(0);
  }
}
```

To install:

```
if (signal(SIGINT, handler) == -1)
{
  printf("Error, unable to install handler!\n");
  exit(1);
}
```

Q5: [5 marks]

Consider the shell variable assignment:

```
set x = ( abc def _123 xyz )
```

1. (1 mark) What is the output of "`echo $#x`"?

   ```
   4
   ```

2. (1 mark) What is the output of "`echo $?x`"?

   ```
   1
   ```

3. (1 mark) What is the output of "`set y = $x; echo $y`"?

   ```
   abc
   ```

4. (2 marks) Show how the word "`_123`" can be removed from `x` using a simple assignment statement. The answer "`set x = ( abc def xyz )`" is **not** acceptable in this case.

   ```
   set x = ( $x[1-2] $x[4] )

   set x[3] = ''
   set x[3] =
   ```

   The last two were accepted, although they leave the element blank as opposed to removing it (`$#x` remains 4)

Q6: [10 marks]

1. (7 marks) Write a shell script `findfile` so that

   ```
   findfile name dir1 dir2 ...
   ```

   searches for the file `name` in the directories specified. If the file is found in one of the directories, the current directory is changed to it. Assume `name` is a file name, and not an absolute or relative path.

```
#!/bin/csh -f
if ( $#argv < 2 ) then
  echo Usage: findfile name dir1 ...
  exit
endif

set cdir = $cwd
set name = $1
shift # remove filename from command line args

foreach dir ( $argv )

  echo Trying $dir
  if ( -d $dir ) then
    cd $dir
  else
    echo $dir not a directory ...
    continue
  endif

  if ( -e $name ) then # check for existence of $name in $cwd
    echo $name found in $cwd
    break
  endif
  cd $cdir # go back to start directory

end
```

This question was poorly done. Many people didn't handle both absolute and relative paths for `dirX`. Many people were using 'C' constructs. Also, many people were using "find", but forgetting that it recursively searches subdirectories, which this script was not specified to do.

2. (3 marks) Why does this script have to be sourced through aliasing rather than invoked as a command? Show an acceptable `alias` command to achieve correct operation for the script.

If this script is invoked as a command, it will be a child process of the calling process (shell). Therefore any change of directory ($cwd) will not be reflected in the parent process.

```
alias findfile 'source findfile \!*'
```

Q7: [10 marks]

Write a C program that takes any number of file names as command line parameters. For each filename the program should create a child process which i) prints the name of the file, and ii) uses `execlp()` to execute the command "`wc -l`" on the file. The parent process should not exit until all the children have exited, and should create the child processes in such a way that they run concurrently, not sequentially. For any children that either a) terminate abnormally, or b) return a non-zero status code, the parent process should print a line indicating either that an abnormal termination occurred, or print the non-zero status code returned.

You may find some of the following function prototypes and macros useful:

```
execlp(char *filename, char *arg0, ... , (char *)0);
int wait(int *status);
int waitpid(int pid, int *status, int options);
void *malloc(long size);
void *realloc(long size);
char *strcpy(char *dest, char *src);
int   strcmp(char *s1, char *s2);
WIFEXITED(status)
WEXITCODE(status)
WIFSIGNALED(status)
WTERMSIG(status)
WIFSTOPPED(status)
WSTOPSIG(status)
```

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

  for ( i=1; i<argc; i++)
  {
    if (fork() == 0)
    {
      printf("%s\n", argv[i]); fflush(stdout);
      execlp("wc", "wc", "-l", argv[i], (char *)0);
      exit(1); /* needed in case execlp() fails */
    }
  }

  i = argc - 1 ;
  while (i)
  {
    int status, pid ;

    pid = wait(&status);
    if (WIFEXITED(status) && (WEXITCODE(status) != 0))
      printf("Child %d exited with return code = %d\n", pid,
             WEXITCODE(status));
    if (!WIFEXITED(status))
      printf("Child %d terminated abnormally.\n", pid);
    i-- ;
  }
}
```

Many people only passed one "wc" to execlp(), or failed to pass "-l" as a separate parameter. Most people didn't make the children run concurrently.

(Use this page for answer overflows)