CSC209S Midterm (L0101)

Spring 1999

University of Toronto

Department of Computer Science

Date:	February 26, 1999
Time:	12:10 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 40 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:	Marking
Student Number:	Scheme
CDF Username:	

Marks:

Q1	10
Q2	5
Q3	10
Q4	10
Q5	5
Total	40

Q1: [10 marks] (1 mark each unless otherwise indicated)

1. What does "`set noglob`" do when typed to the shell?
   Turns off filename expansion

2. (2 marks) List two different ways in which a shell script can protect itself from unexpected behaviours due to aliased commands.
   (1 mark) `#!/bin/csh -f`
   (1 mark) preface commands with '\' to over-ride any aliases

3. Explain the difference between absolute and relative pathnames.
   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)

4. What is the difference between '`mv file1 file2`' and '`cp file1 file2; rm file1`'?

   The first merely changes the name in the directory listing (no new file is created, and the inode for the file remains unchanged). The latter creates a new file (with a new inode) and then deletes the first file.

5. What is the difference between the `stat()` and `lstat()` functions?

   When used on a symbolic link, stat() returns the inode information for the file the link points at, whereas lstat() returns the inode information for the file containing the link information.

6. The directories for searching executables are specified both in the environment variable `$PATH` and in the local shell variable `$path`. Which is the appropriate way to add `$HOME/bin` to the search path?

   ```
   setenv $PATH $HOME/bin:{$PATH}
   ```

   or

   ```
   set $path = ($HOME/bin $path)
   ```

   `Either is fine. Setting one automatically changes the other ($path is a "special shell variable").`

7. (3 marks) Explain the difference between a "hard-link" and a "soft link". Give an example of using each.

   A hard-link is just another directory entry which points at a file's inode. A symbolic link creates a new file with string containing the path (absolute or relative) of the "linked" file. To create a hard link: "ln file1 newLinkHard". To create a symbolic link, "ln -s file1 newLinkSoft".

Q2: [5 marks]

Write a code fragment which takes a string defined as

```
char *cmdLine ;
```

and processes it to put it in the correct format for `execvp(char *filename, char *argv[]);` You are to use `strtok(char *s, const char *delim)` to break the string into tokens. You can assume that there are no more than MAX_CMDLINE_ARGS tokens to be processed. Define whatever new variables you need.

```
int   i = 0 ;
char *argv[MAX_CMDLINE_ARGS+1];

memset(argv, 0, sizeof(argv));

argv[0] = strtok(cmdline, WHITESPACE);
while (argv[i] != (char *)NULL)
{
  i++ ;
  argv[i] = strtok(NULL, WHITESPACE);
}
```

Q3: [10 marks]

a) (2 marks) "`ls -l`" lists files in the format

```
-r-xr-xr-x  3 bin      89564 May 3  1996 /bin/sh
```

where the size of the file is the 4th word in the line. How can you assign the size of a file to a shell variable using "`ls -l`"?

```
set temp = `ls -l $fileName`
set size = $temp[4]
```

or

```
set size = `ls -l $fileName | awk {print $4;}`
```

b) (7 marks) Write a shell script "`dirsize dir`" that computes the total size in bytes of all files in the directory `dir`. Do not count the size of the directory entries (this includes, but is not limited to, "`..`" and "`.`").

```
@ bytes = 0
foreach file ( * )
  if (! -d $file)
    set temp = `ls -l $file`
    @ bytes = $bytes + $temp[4]
  end if
end
echo "Total size = $bytes bytes."
```

c) (1 mark) Does the result calculated accurately reflect the total bytes of disk space occupied by these files? Explain.

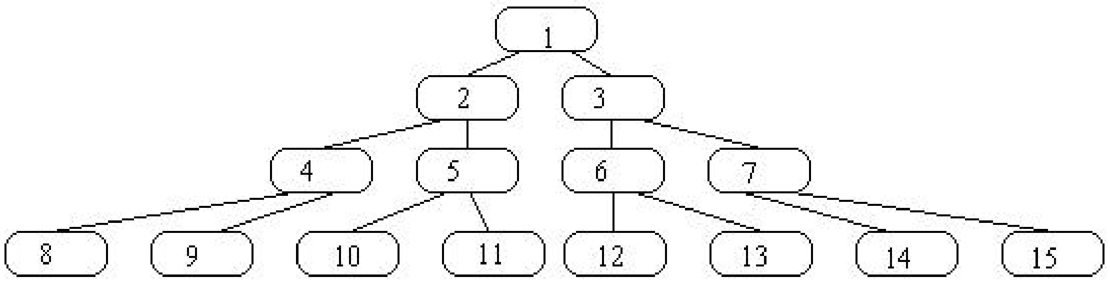
It may not be accurate if there are hard or soft links in the directory. For example, if `name1` and `name2` are both hard links pointing to the same file, then that file's space will get counted twice. To avoid this have your script check inode numbers and throw out duplicates.

Q4: [10 marks]

Write a program that takes a single integer argument *n* from the command line and creates a binary tree of processes of depth *n* (*i.e.* a group of processes descended from a common ancestor in a tree-like manner). When the tree is created, each process should display the phrase "I am process *x*, my process ID is *y* and my parent's is *z*" where *x* is the number of the process (see diagram below), create any necessary child processes, wait for them to terminate, and then terminate. The nodes of the process tree should be numbered according to a breadth-first traversal (*e.g.* see diagram below). For example, if the user enters the command

```
% tree 4
```

then the process tree would look like



and the output would be (something like)

```
I am process 1, my process ID is 465 and my parent's is 321.
I am process 2, my process ID is 466 and my parent's is 465.
⋮
I am process 15, my process ID is 499 and my parent's is 492.
```

Make sure the original parent process (#1) does not terminate until all of its children have died.

```
int main(int argc, char *argv[])
{
  int procNum = 1 ;
  int child[2] ;

  int numChildren = 0 ;
  int status ;
  int maxDepth ;

  assert(argc == 2);
  maxDepth = atoi(argv[1]); /* convert depth parameter */

  while (numChildren < 2)
  {
    if (numChildren == 0)
      printf("I am process %d, my process ID is %d"
             " and my parent's is %d.\n",
              procNum, getpid(), getppid());
    child[numChildren] = fork();
    if (child[numChildren] == 0)
    {
      numChildren = 0 ;
      procId *= 2 ;
      if (! --maxDepth) exit(0);
    }
    else
    {
      numChildren++ ;
      procNum++ ;
    }
  }

  while (numChildren) wait(&status);
}
```

Q5: [5 marks]

Write a C function `int IsPalindrome(char *s)`. This function returns 1 if the strings identified by `s` is a palindrome and 0 otherwise (a string is a palindrome if reversing it results in the same string). Assume `s` is null-terminated and do not make a copy of `s`.

```
int IsPalindrome(char *s)
{
  char *start = s, *end = s;

  if (! *s) return 1; /* an empty string is a palindrome */
  while (*end++); end-- ; /* position end of string */

  while (*start == *end)
  {
    start++ ;
    end-- ;
    if (start > end) break ;
  }

  if (start > end) return 1 ;
  else             return 0 ;







}
```