CSC209S L0101/L5101 Spring 00

Assignment 2

Jan 27th, 2000 Due February 25th, 2000

Question 1 (30 Marks)

Write a "mini-shell". It will be invoked with the command "`miniShell`", and should behave as follows:

1. When the program begins, it scans all directories in the `$PATH` environment variable looking for executable files, and adds the name and absolute path of each one it finds to a hash table (the name should be used as a "key" for looking up the absolute path). You are responsible for designing and implementing the hash table yourself.
2. Your program will then repeatedly display the following prompt: "`msh> `" and accept simple commands of the form "`<commandName> [<parm>]`" where `<commandName>` is the name of an executable to run, and `[<parm>]` is a list of 0 or more parameters to pass to the executable as command line parameters.
3. If the command cannot be found, your program is to output "`<commandName>: command not found.`", and then display a new prompt on a new line.
4. The user of your program is to terminate it by typing `^D` to signal `EOF`.

Your program is to use `fork()` and then one of the `exec()` family of functions to execute the desired command. If `<commandName>` is not an absolute or relative path, then you will attempt to find an absolute path for it by looking it up in your hash table. You are not required to handle pipes or I/O redirection, nor are you required to do a command history mechanism, filename expansion, or aliasing. "`miniShell`" does not run processes in the background, so after `fork()`-ing to execute a command, it should wait for the child process to terminate before displaying a new prompt.

Hand in a printed version of your csh script, as well as submitting it electronically on CDF using "submit -N a2 csc209h miniShell.c". You can overwrite a previous submission by adding the "-f " switch to the submit command. **Note:** *The file you submit must be named "miniShell.c" or else the electronic marking program will not find it, and you will get a mark of 0.*

Question 2 (20 marks)

Write a program "ttar" (trivial tar) to concatenate files into an "archive". Its use is one of the following:

```
% tar -c <tarfile> {<file>}
% tar -l <tarfile>
% tar -x <tarfile>
```

where `<tarfile>` is a non-optional parameter giving the name of the file the data is to be written into or read from, and `{<file>}` is a list of one or more file/directory names. When invoked with the `-c` option your program creates the named `<tarfile>`, and adds the named `<file>`s to it. If a name to be added is that of a directory, then `ttar` will recursively add the contents of that directory to `<tarfile>`. The format for the file is shown below. When invoked with the `-l` option, it gives a listing of all files stored in `<tarfile>`, and when invoked with the `-x` option, it extracts the files. Note that when files are extracted, the directory structure must be preserved.

The ttar file structure is as follows:

`"ttar1.0"`	Bytes 0-6
plus one or more of the following records	
<length of file #1 as an unsigned long>	Next sizeof(unsigned long) bytes
<relative pathname of file #1 - zero-terminated character string>	Next strlen(pathname)+1 bytes
<file #1 data>	Next "size of file" bytes

Your program is required to do any necessary error checking. That includes, but is not limited to, the following list:

- unable to read named file
- unable to write named file
- unable to create `<tarfile>`
- unable to create required directory
- `<tarfile>` file named does not exist (`-l`, `-x` options only)

Hand in a printed version of your csh script, as well as submitting it electronically on CDF using "submit -N a1 csc209h ttar.c". You can overwrite a previous submission by adding the "-f " switch to the submit command. **Note:** *The file you submit must be named "ttar.c" or else the electronic marking program will not find it, and you will get a mark of 0.*