Statement: Develop a few Unix systems utilities for process and filesystem management

Objectives: Practicing Unix system calls, understanding Unix process and filesystem management

Assignment Description

In this assignment, you will develop three simple Unix systems utilities (programs) for process and filesystem management: mytree.x, mytime.x, and mymtimes.x. The details of these utilities are described below:

- mytree.x: print files (and subdirectories) under a given directory in a tree-like format. Syntax: mytree.x [dir], where dir is an optional command parameter, which specifies the starting directory that the command will work on. If dir is not given, use the current working directory as the beginning directory. From the starting directory, mytree.x needs to descend to into subdirectories, if any such subdirectories exist. The output of mytree.x should be similar to that of the Unix command *tree*. That is, you should align the subdirectories and files based on their hierarchical positions in the filesystem. You can print the horizontal and vertical lines using concatenated "-" and "|", respectively, instead of real-lines as in the output of the Unix tree command. You do not need to sort the files and directories in the output as done in the Unix command tree. An example run of implemented mytree.x is given at the end of the assignment description.

- mytime.x: run a command and track its running time. Syntax: mytime.x cmd [arguments], where cmd is the command to run, and the arguments are the optional command line arguments to the command cmd. The output of mytime.x should be similar to the Unix command *time*. At the minimum, mytime.x should report three values: 1) user CPU time; 2) system CPU time; and 3) elapsed wall-clock time for running the command cmd. There are different ways to implement this utility. As a starting point, you can read the manual pages of the Unix system calls wait, waitpid, wait3, wait4, getrusage, and times. You can assume that the "cmd" command-line argument of mytime.x is in one of the search paths included in the PATH environment variable.

- mymtimes.x: report the hourly number of files modified in the last 24 hours. Syntax: mymtimes.x [dir], where dir is an optional command parameter, which specifies the beginning directory that mymtimes.x will work on. If dir is not given, use the current working directory. For the given beginning directory, this command will examine the last modification time of all the regular files (under the given directory and all the subdirectories, if any). It will group the files according to their last modification time into hourly time intervals in the last 24 hours, and report the number of files in each hourly time interval in the last 24 hours. An example run of mymtimes.x is provided at the end of the assignment description.

You cannot use the "system()" library routine (the function "system") in this assignment, or in any other similar ways to directly run the corresponding Unix command (such as time and tree) to implement these programs. You can use fork/exec to run the "cmd" in the implementation of mytime.x.

Grading Policy

A program with compiling errors will get 0 point. A program containing "system()" function will get 0 point. Total points: 100.

1. proper makefile file, compilation and run without runtime errors (10)
2. command "mytree" (30)
3. command "mytime" (30)
4. command "mymtimes(30)

Assignment Submission

Tar your makefile, all source code files (and README file if any) into a single tar file. Make sure that you tar all the files successfully (you can check the content of the tar file by running "tar -tvf tar_file"). You are responsible for empty tar file or wrong tar file submitted, and late penalty will apply if you need to re-submit after the deadline. When you tar your files, please do not use any compress related options such as j or z; it will complicate our efforts in automating the extraction of files from the tar file.

Here is a simple example on how to use the Unix command tar (assuming all your files related to this project are under one directory, say proj1). To tar all the files:

```
tar -cvf proj1.tar *
```

To check the contents in the tar file:

```
tar -tvf proj1.tar
```

To untar a tar to get all files in the tar file (to avoid potential overwriting problems, it is better to do this under a different directory instead of the original proj1 directory):

```
tar -xvf proj1.tar
```

```
shab@linprog2 (~...assignments/proj1) % mymtimes.x
~shab/courses
Wed Jan 27 11:20:11 2021: 0
Wed Jan 27 12:20:11 2021: 0
Wed Jan 27 13:20:11 2021: 0
Wed Jan 27 14:20:11 2021: 0
Wed Jan 27 15:20:11 2021: 0
Wed Jan 27 16:20:11 2021: 0
Wed Jan 27 17:20:11 2021: 0
Wed Jan 27 18:20:11 2021: 0
Wed Jan 27 19:20:11 2021: 0
Wed Jan 27 20:20:11 2021: 0
Wed Jan 27 21:20:11 2021: 0
Wed Jan 27 22:20:11 2021: 0
Wed Jan 27 23:20:11 2021: 0
Thu Jan 28 00:20:11 2021: 0
Thu Jan 28 01:20:11 2021: 0
Thu Jan 28 02:20:11 2021: 0
Thu Jan 28 03:20:11 2021: 0
Thu Jan 28 04:20:11 2021: 0
Thu Jan 28 05:20:11 2021: 0
Thu Jan 28 06:20:11 2021: 0
Thu Jan 28 07:20:11 2021: 0
Thu Jan 28 08:20:11 2021: 0
Thu Jan 28 09:20:11 2021: 1
Thu Jan 28 10:20:11 2021: 3

shab@linprog6 (~...assignments/proj1) % mytree.x test1
test1
|--- example2.c
|--- example1.c
|--- test2/
|    |--- example23.c
|    |--- test3/
|    |    |--- example31.c
|    |--- example32.c
|--- test1/
```