COP5570 Programming Assignment 1 (Warm-up)
Poor Man's Parallel Program Launchpad

OBJECTIVES

- Get familiar with the UNIX programming environment
- Experience with distributed systems
- Use the *make* utility

DESCRIPTION

Concurrent, parallel, and networked applications often require a program to be executed on different machines. These instances of the program on different machines can then work together to solve problems. In this assignment, you will write two utility programs, *myprun* to launch the program on different machines and *mypkill* to kill (collaborative) processes on different machines.

Myprun

The `myprun` command reads from a file (machine file) the set of machines to run the program, takes the number of processes (*np*) and a C program source file as arguments, and starts the program on different machines. The command format is as follows

myprun [options] <program> [options]

The options can be nothing, '`-f mf`', or '`-np #`'. '`-f mf`' replaces the default machine file with file `mf`. If this option is not presented in the command, the default file `machinefile` will be used by the command. The number following the '`-np`' flag is the number of processes to be launched. If this option is not presented, the default `np` is 1. The `program` in the command is the path of a C source file.

The machine file consists of lines of 'username@machinename'. For example, the content of a machine file can be as follows. For the program to work correctly, you must make sure that you can run command on that machine with that username.

```
xyuan@linprog1
xyuan@program1
xyuan@linprog2
xyuan@crux
```

The number of processes can be anywhere from 1 to 15. The list of machines will be used in a round-robin fashion. For example, Command

```
./myprun -f machinefile -np 2 test.c
```

will start an execution of test.c on linprog1 and the second one on program1 assuming the `machinefile` content is shown above. Command

```
./myprun -f machinefile -np 5 test.c
```

will start 5 programs; each of the first four will be on one of the machines in the file; the fifth program will start on linprog1. Before the program is executed, you must set two environment variables to be passed into the program: TSIZE=np and MYID = 0..np-1 (one value for each process). These two variables will allow the program being executed to behave differently so as to collaboratively solve problems. Some examples of the myprun command are:

```
<diablo:1001> ./myprun test.c
<diablo:1002> ./myprun test.c -f mf -np 2
<diablo:1003> ./myprun -f mf test.c -np 3
<diablo:1004> ./myprun test.c -np 2
<diablo:1005> ./myprun -np 20 test.c
<diablo:1005> ./myprun test.c -f mf
```

Mypkill

The `mypkill` command reads from a file (machine file) the set of machines where the processes will be killed, and kills processes specified by a pattern on each of the machines (using the UNIX `pkill` utility on each machine). The command format is as follows:

mypkill [options] <pattern> [options]

The options can be nothing or '`-f mf`'. '`-f mf`' replaces the default machine file with file `mf`. If this option is not presented in the command, the default file `machinefile` will be used by the command. The `pattern` is a substring in the command line for the process to be killed (using 'pkill pattern'). Some examples of the `mypkill` command are:

```
<diablo:1001> ./mypkill a.out
<diablo:1002> ./myprun a.out -f mf
<diablo:1003> ./myprun -f mf a.out
```

REQUIREMENT

- The commands can launch/kill processes from both `linprog` and `program` machines.
- You must handle the cases that the machines used to run commands may or may not share the file system.
- `Myprun` must launch the commands on different nodes concurrently (as opposed to one after another).
- If you create any temporary files, you must clean up (remove) the temporary files when your program stops running.
- You can assume (1) that all machines have the `gcc` compiler, (2) that the program.c can be compiled with 'gcc program.c', and (3) that the executable runs with no command line arguments.
- You can assume all user inputs are correct. If the user inputs are wrong, the program can have non-deterministic behavior.

GRADING POLICY

This program should be developed on `linprog` and `program`. A proper makefile should be used in this project: (1) Command 'make' should produce the two executables; (2) Command 'make demo' should run the program (use the existing executable if it has been generated or produce the executable if it is not available); (3) Command 'make clean' should remove .o files and executables from the directory; and (4) the program should be compiled with '-Wall -ansi -pedantic' or '-Wall -std=c99 -pedantic' flags. Compiling the program with flags should not have any warning message. You should also write a README file that tells (1) how to compile and run your program and (2) how to repeat your demo.

In the grading, each item will be graded in three steps: 0 point, half of the points, and full points.

- Proper README and makefile (10 pts, half for each of linprog and program)
- 'myprun' functionality (22 points, half for each type of machines)
- 'mypkill' functionality (22 points, half for each type of machines)
- Properly handle all different types of command line arguments (10 points)
- 'myprun' handles a shared/non-shared file system (10 points)
- 'myprun' must start the command on different nodes concurrently (6 points)
- 'myprun' sets the two environment variables (4 points)
- Clean up temporary files (6 points)
- Demo and project submission (10 pts). You must design your demo to show all features in your program WITHIN THE GIVEN TIME. Failing to do so will lose not only the demo points, but also the points for the features that you do not have time to demo. You cannot touch your source code (unless you are asked to) during (and after) demo.
- **-20 points – unknown bug of any kind caught during/after the demo**
- **-10 points – first unknown error/unimplemented feature**
- **-40 points – the code has to be touched in any way after demo starts**
- -5 points – each warning message.
- Reporting a bug in the sample executable gets extra 5 pts.

DEADLINES AND MATERIALS TO BE HANDED IN

Deadline: May 29. The starting of the demo is the due time.

- Project Demo (15 mins)
- Submit the hard copy of your code to the TA during project demo
- Email the TA your whole program package that can repeat the demo before your demo.

MISCELLANEOUS

- All programs will be checked by an automatic software plagiarism detection tool.
- This is a small program, about 150 lines in the sample implementation.
- This project will account for 6% of the course numerical grade.
- Start early!!