50.005 Computer System Engineering

Programming Assignment (Project) #1: Process Tree Management

Due date: end of recess week (Sun 13 March, 11:59 PM)

You can work in a group of max 2 students

1. Purpose

- To develop a processmgt.c / processmgt.java program by using processes (fork, exec, wait, system) in C or (ProcessBuilder) in java to execute a graph of user programs in parallel
- Graph of user programs are executed by reading commands in text file
- To model control – and data – dependencies
- Control dependence: a program cannot start until its predecessor(s) is finished
- Data dependences: a program requires input from its predecessor(s) before it can execute
- Example: make creates a dependence graph for all the files, independent files are compiled individually while dependent files have to wait for all its control dependencies before running

2. Description

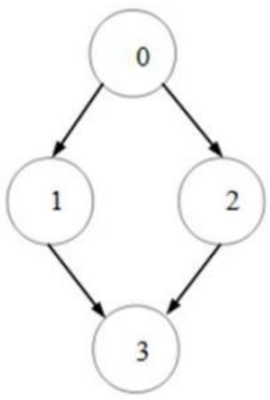

Figure 1: Example graph

- To analyse a graph of user programs, determine which programs are eligible to run and running programs that are eligible to run that moment in time

- According to Figure1, node 0 can be executed first because it is not a child of any other node
- After node 0 finishes executing, then node 1 and node 2 can be executed in parallel (they both should be started immediately without waiting for either 1 or 2 to finish
- Only after both 1 and 2 finish, the final node 3 can be executed
- Each node in the graph represents one program to be executed. Each node will contain:
- The program name with its argument;
  - "Pointers" to child nodes;
  - The input file to be used as standard input for that program;
  - The output file to be used as standard output for that program
- Only when all of its parent nodes (nodes that contain a pointer to this node) have completed their own execution, a node has the right for execution
- The processmgt.c / processmgt.java will fork and exec each of the nodes as they become eligible to run
- IO redirection must be used so that each node can get its standard input from a file and write its output to a file
- Should handle multiple root nodes which can be executed simultaneously

3. Process Tree Specification

- Run processmgt.c program as follow in terminal, $./processmgt testproc.txt
- A text file (eg. testproc .txt) will represent the structure of the graph. This program takes as input a text file entered by the user at the terminal prompt. It then reads this file line by line parsing out information (colon delimited). Each node will be represented by a line in the following format:
- < program name with arguments:list of children ID's:input file:output file >
- For example, in your testproc.txt, a few command lines are put line by line :

```
echo hi there:1:stdin:echo-out.txt
echo wazzup:2 3:stdin:echo2-out.txt
cat:4:echo-out.txt:cat-1.txt
cat:4:echo2-out.txt:cat-2.txt
cat cat-1.txt cat-2.txt:none:stdin:sink.txt
```

- NOTE: If there are no children for a node, it must be put as "none" . If the input and output streams to a command are not redirected and they use the default STDIN and STDOUT streams, they are specified using "stdin" and "stdout" in the input and output file.
- The nodes will implicitly be numbered from 0 to (n-1) from the order the nodes appear in the text file (where n = total number of nodes in the graph). The children IDs will correspond to this numbering system. The ID numbers are used mainly to create pointers from parent nodes to children nodes.

4. Node Structure

- Code for suggested node structure:

```
// for 'status' variable:
#define INELIGIBLE 0
#define READY 1
#define RUNNING 2
#define FINISHED 3

typedef struct node {
      int id; // corresponds to line number in graph text file
      char prog[MAXLENGTH]; // prog + arguments
      char input[MAXLENGTH]; // filename
      char output[MAX_LENGTH]; // filename
      int children[MAX_CHILDREN]; // children IDs
      int num_children; // how many children this node has
      int status; // ineligible/ready/running/finished
      pid_t pid; // Process id when it's running
}  node_t;
```

5. Overview processmgt.c / processmgt.java

- The processmgt.c program will first parse the text file (eg. testproc.txt) in the first argument
- The program then will construct a data structure that models the process tree management
- The program will start executing the nodes
- Important points in the program :
    - Determine which nodes are eligible to run
    - Execute those nodes
    - Wait for any node to finish
    - Repeat this process until all nodes have finished executing
- A sample text file is given to you as input of your program and a sample output file is given to you to compare your results with sample output

6. Useful System Calls and Functions

- In C :
    - fork, dup2, execvp, wait, strcmp
- In Java:
    - BufferedReader, String, ProcessGraph, ProcessGraphNode, ProcessBuilder, Process, Thread, ProcessBuilder.redirectInput(), ProcessBuilder.redirectOutput(), Process.waitFor()

7. Hinst and Error Handling

Hints:

- implement a function which marks the nodes that are ready to run
- A sample parsing function are provided. The function takes as input one line of the input file and a pointer to a node_t structure.
- It populates the prog, input, chidren and num_children fields of the node and returns a value of 1 on success.

Error Handling:

- To check the return value of all system calls in processmgt.c / process.java for error condition
- To make sure the proper number of arguments are used when it is executed
- A useful error message should be printed to the screen if there were an error
- Program should be able to recover from errors if possible
- In C :
    - use perror() function to print error messages and exit the program
- In java:
    - Try and catch exception error

8. Documentation

- Include a README file which describes your program. It needs to contain the following:
    - The purpose of your program
    - How to compile the program
    - What exactly your program does
- The README file does not have to be very long, as long as it is understandable for first time user without any confusion.
- Within your code you should use one or two sentences to describe each function that you write.
- At the top of your README file and main C source file please include the following comment:

```
/* Programming Assignment 1
* Author : Full Name
* ID : Student ID
* Date : dd/mm/yyyy */
```

- If you are working in a group, write the group members names and IDs, and only one of you need to submit the project.

9. Grading

- 5% README file
- 20% Documentation within code, coding, and style (indentations, readability of code, use of defined constants rather than numbers)
- 75% Test cases (correctness, error handling, meeting the specifications)
- Please make sure to pay attention to documentation and coding style. A perfectly working program will not receive full credit if it is undocumented and very difficult to read.
- The test cases will not be given to you upfront. They will be designed to test how well your program adheres to the specifications. So make sure that you read the specifications very carefully. If there is anything that is not clear to you, you should ask for a clarification.
- This is a group programming assignment. Collaboration between maximum two students is permitted and detected cheating between groups will automatically result a Zero for this assignment.