Singapore University of Technology and Design

Established in collaboration with MIT

Computer System Engineering
50.005
Dr. David Yau

OS Programming Assignment
Process Tree Management

Contact us
dima_rabadi@mymail.sutd.edu.sg
jie_yang@mymail.sutd.edu.sg
liza_ng@alumni.sutd.edu.sg

The Goal of this project

➔ **To execute a group of processes that have control and data dependencies between each other.**

Control dependencies: a process cannot be started until another process finishes.

predecessor and successor

Data dependencies: a process requires input from another process before it can start.
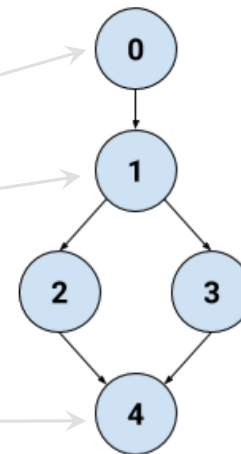
cat file.txt

Example

Input file format:

<program>:<child nodes>:<input>:<output>

Example:

```
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
```


Each **node** in the graph represents a *process*, and the **edges** represent *dependency relations* between processes.

e.g. Process 1 can only start after Process 0 finishes. After Process 1 finishes, Processes 2 and 3 can be run in parallel.

Steps

1. **Parse the text file** containing the graph of processes.

2. **Execute the processes in the correct order**, such that dependency relations between processes are properly met.

Useful functions (C)

Parsing of input file: **strtok()**

Process tree representation: **node** struct

```
typedef struct node {
    int id;// corresponds to line number in graph text file
    char prog[MAX_LENGTH];// prog + arguments
    char input[MAX_LENGTH];// 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;
```

Process execution: **fork()**, **exec()**, **dup2()**, **waitpid()**

- **fork()** can be used to create a new process, and **exec()** to run a program within the newly forked process.

- **dup2()** can be used to redirect the input and output for a process.

- **waitpid()** can be used to wait for a process to finish executing.

Useful classes and methods (Java)

Parsing of input file: **BufferedReader**, **String**

Process execution: **ProcessBuilder**, **Process**, **Thread**

➜ **ProcessBuilder.redirectInput()** and **ProcessBuilder.redirectOutput()** can be used to set the input and output file for a process

➜ **Process.waitFor()** can be used to wait for a process to finish executing.

Input/Output Redirection

- You can choose any possible way to redirect you input and output, suggested examples:
  - system (cat file1.txt > file2.txt)
  - "ls -l | wc -l"
  - dup2 in C
  - ProcessBuilder.redirectInput() and ProcessBuilder.redirectOutput() in Java

Instructions

- Download the assignment package from eDimension.
  - The package includes the instruction handout and sample input and output files to test your code.

- Assignment weightage: 5% of final course grade

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

- Submit your Java/C source code to eDimension, along with a README file, your name and ID.