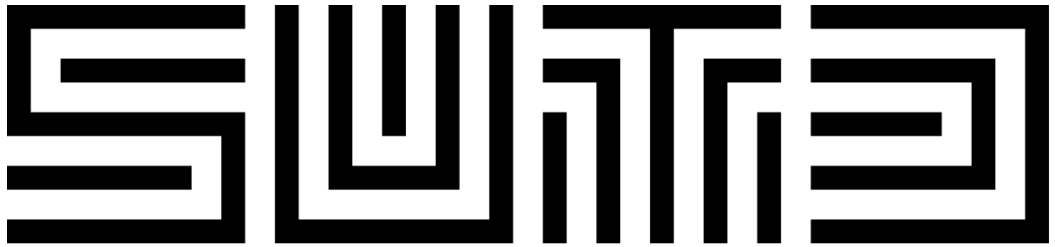


Computer System Engineering
50.005

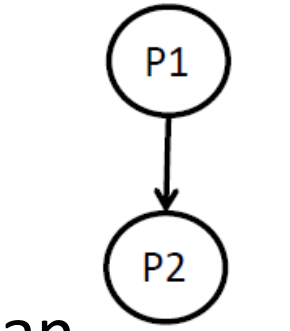
Dr. David Yau and Dr. Jit Biswas

Operating Systems: Programming Assignment
Objective: Process Tree Management

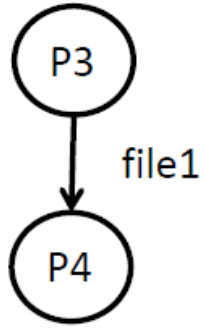
1/02/2018

The Goal

- To execute a group of processes that have control and data dependencies between each other.
- **Control dependency:** A process must terminate before successor can begin,
  - eg. P1 gives waits for time to start (let's say 3pm), and terminates
  - P2 starts only when P1 has terminated
- **Data dependency:** A process requires input from another process before it can start
  - eg. P3 produces output file1 and P4 takes input from file1



Control dependency



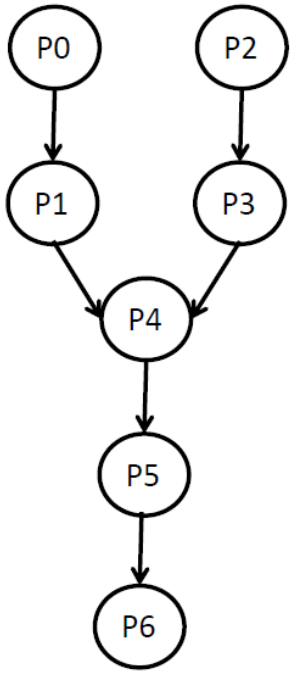
Data dependency

Example

- A graph-file represents structure of the graph
- Each **node** in the graph represents **a process**, and the **edges** represent **dependency relations** between processes
- Its represented in following format:
- **Input file format:**
  - <program name with arguments> : <child nodes IDs> : <input> : <output>

- **Example of graph file:**

```
sleep 10:1:stdin:stdout
echo "Process P1 running. Dependency to P4 is cleared.":4:stdin:out1.txt
sleep 15:3:stdin:stdout
echo "Process P3 running. Dependency to P4 is cleared.":4:stdin:out2.txt
cat out1.txt out2.txt:5:stdin:cat-out.txt
grep 3:6:cat-out.txt:grep-out.txt
wc -l:none:grep-out.txt:wc-out.txt
```



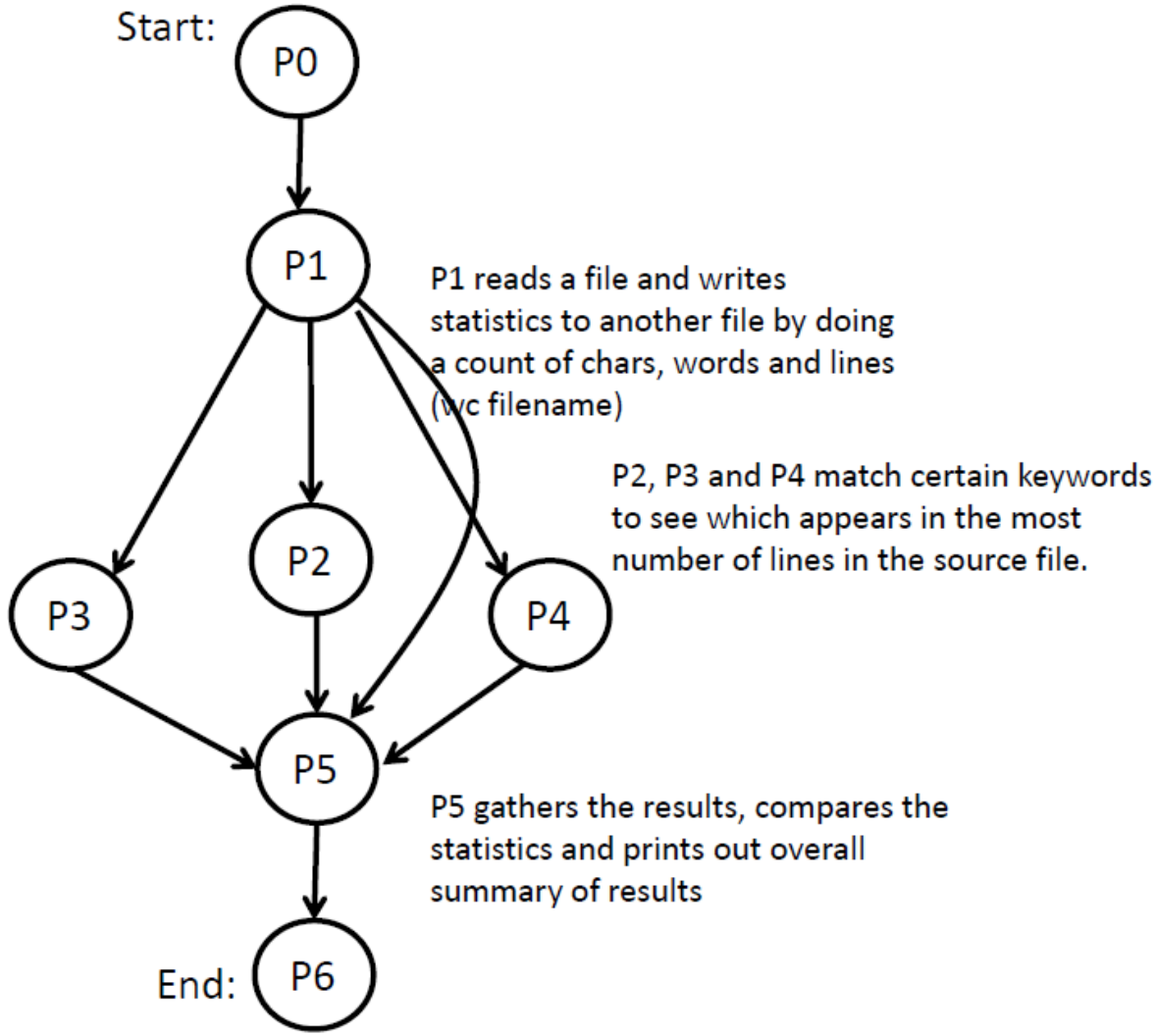
Eg. Process 1 can only start after Process 0 has finished. After Process 1 finishes, Processes 2 and 3 can be run in parallel.

Example

- Graph: Processes with **multiple child**
  - Represented by **space delimited** IDs

- **Example of graph file:**

```
sleep 10:1:stdin:stdout
wc ATaleOfTwoCities:2 3 4 5:stdin:out1.txt
grep Paris:5:ATaleOfTwoCities:outParis.txt
grep London:5:ATaleOfTwoCities:outLondon.txt
grep city:5:ATaleOfTwoCities:outcity.txt
wc outcity.txt outParis.txt outLondon.txt:none:stdin:sink.txt
```



Steps

- Parse the text file containing the graph of processes.
- 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 *args[MAX_LENGTH/2 + 1];
int num_args;
char input[MAX_LENGTH]; // filename
char output[MAX_LENGTH]; //filename5
int parents[MAX_PARENTS]; //parents's IDs
int num_parents;
int children[MAX_CHILDREN]; //children's IDs
int num_children;
int status; // ineligible /ready/ running/ finished
pid_t pid; //process id when it is runnning
} node_t;
```

- Process execution: fork(), exec(), dup2(), wait()
  - 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
  - wait() can be used to wait for any process to finish executing

Useful classes and methods (Java)

- Parsing of input file: BufferReader, 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 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
- Decide which language do you prefer based on your background
  - Java or C language
- Read the tasks one by one and use the starting code provided on eDimension
- Assignment weightage: 5% of final course grade
- Due date: end of recess week (**9th March, 11:59pm**)
- Don’t hesitate to ask for help from the instructors in the lab!
- Complete the assignment with features and upload the Java or C program along with README file, your name and ID to eDimension before the above due date.