CMS 330, Spring 2019, Midterm Study Guide

Topics:

* Process from going from source code (high level language) to machine code (low level language).
* 4 components of a grammar
* Apply specific grammar to produce specific expressions
* Draw parse trees given a specific grammar
* Draw transition diagrams to differentiate stream of characters into discrete tokens (ala your work and code on project 1)
* Explain top-down parsing.
* What is backtracking and why do we need it during parsing?
* Why is what we did called “predictive” parsing? How is prediction used in the process?
* Given a list of tokens, under what conditions does our parser generate an error? (There are multiple.)
* We say that the OS virtualizes the system resources. What does this mean, and what are the goals of virtualization?
* What is Limited Direct Execution?
* Pros and Cons of OS as library vs. System Calls
* Why do we need the concept of privileged operations?
* What happens when process wants to do a privileged operation?
* Why is it desirable to separate mechanism from policy in an OS?
* Context switches, changing state of a process, tracking processes (and their states), what to do about slower operations?
* Scheduling algorithms: FIFO, SJF, STCF, Round Robin, Multi-level Feedback Queue
* Calculate turnaround time for a given scheduling algorithm given the specifics for a set of processes.

Practice Problems: You should NOT assume these problems cover the totality of possibilities!

1. Parse tree/CFG problems from class: <https://spr19-rollins-cms330.github.io/resources/cfg-problems.docx>
2. Write a context-free grammar (not a regular expression!) for variable names that may begin with a lowercase letter or underscore character, followed by any number of lowercase letters, uppercase letters, digits, or underscore characters.
3. Consider the following context-free grammar:

E -> T E'

E' -> + T E' | empty

T -> F T'

T' -> \* F T' | empty

F -> (E) | digit

digit -> 0 | 1 | 2 | ... | 9

Use the LL parser technique (what we did in class) to construct parse trees for the following expressions:

* 1. 1 \* 2 + 3 \* 4
  2. (5 + 6) \* 7
  3. (8 \* 9) + (0 \* 1)

1. Draw a transition diagram that recognizes and differentiates amongst the following tokens:

+, +=, ++, =, ==

1. When a system call occurs, the hardware will redirect execution to special trap handler code in the OS. The OS must set up a trap table to inform the hardware of the location of all interrupt-handling routines. Explain when and how the trap table is initialized and who is responsible for the initialization. How does the system prevent regular user processes from modifying the trap table once it is installed?
2. Draw the diagram showing the flow of execution for a system call.
3. Suppose that a system uses a round-robin scheduling discipline. The system is initially empty, then the following sequence of jobs arrives
   * Job A arrives at time 0 and has a service requirement of 4
   * Job B arrives at time 0 and has a service requirement of 3
   * Job C arrives at time 2.5 and has a service requirement of 3
   * Job D arrives at time 3 and has a service requirement of 2

The system keeps the jobs in alphabetical order. When it's time to make a scheduling decision, the scheduler always chooses the next available job in the alphabetical listing. The quanta size is 1.

1. Draw a scheduling diagram showing the order in which the jobs execute. Indicate the arrival time and departure time of each job.
2. Calculate the average turnaround time.