CS2303 Systems Programming Concepts A14

Lab 3 {August 25, 2014} 5 points

Command Line Arguments and Event List Operations

Programs 3 and 5 involve event-driven simulation. The **key** data structure in event-driven simulation is a linked list known as an event list. Lab 3 introduces event list operations, working with **structs** and **pointers** and inputting parameters to a C program using Linux command line arguments.

Your Lab 3 program begins by reading in one command line argument sources. Thus, the form is:

```
./lab3 sources
```

and an examples is:

```
./lab3 10
```

Assume lab3 reads in sources lines of input from a script file where each line of input takes the form:

```
process_id   arrival_time   cpu_time   io_time
```

where arrival_time, cpu_time and io_time are in simulation units (i.e. 100 milliseconds)

For example, an input line:

```
20166  30  17  205
```

indicates that process **20166** arrives at the scheduler at simulated time **3.0** seconds needing **1.7** seconds of CPU service and **20.5** seconds of I/O service. For each line of input, your program needs to create a node struct corresponding to that process. Note – make no assumptions about the order of the input lines read in.

For Lab 3, you must create at least three functions that operate on an event list: add_event, get_nextevent, and print_eventlist.

add_event

Assume an event list built as a linked list where events are stored as structures in **chronological order** (based on process arrival time) such that the pointer to the event list points to the struct that is the first event to occur in simulated time and the last node on the linked list chain (before the **NULL** pointer) points to the struct that holds the event that will occur the farthest into the future of all the events on the event list.

The **add_event** function receives a pointer to the node structure created in conjunction with an input line and a pointer to the event list and inserts this new event node into the event list in the proper place based on the node's arrival_time.

get_nextevent

The **get_nextevent** function uses a pointer to the event list to return a pointer to the next event to occur in simulated time (i.e., the first node in the event list) and deletes this event from the front of the event list.

print_eventlist

The **print_eventlist** function uses a pointer to the event list to print out all the information for each event on the event list in chronological order.

As good programming/debugging technique, you should also write two other help functions, **create_node** and **print_node**. While these two functions are NOT required for lab 3, you should find them useful for debugging Program 3 and they are recommended for Program 3 but NOT required for Lab 3.

Lab 3 main program

The Lab 3 main program uses the sources command line argument to read in sources input lines. For each input line, it creates an event list node and then adds this node to the event list using add_event. Once the event list is completely built, your program prints it out using print_eventlist. Next, main takes the first event off the event list using get_nextevent, prints out its contents, takes the second event off the event list, prints out its content and finally prints out the current remaining event list.

Lab 3 Assignment

0. Prior to coming to the lab prepare a preliminary solution to the assignment above.
1. Create a make file.
2. Test the program under your own test data input either from the terminal or a file.
3. Run the program on the provided test data file 'lab3.dat' redirecting the output to eventlist.txt.
4. Create a README file that contains any useful information to assist in the grading of your lab program.
5. Create a tarred file that contains all the source and header files, the make file and the README file and your output file.
6. Use the Linux version of 'turnin' to turn-in the tarred file. [The deadline for all lab turn-ins is 24 hours after the beginning of your assigned lab.]