CPU Scheduling

CSC 213 – October 4, 2021
Agenda for Today

1. Bug of the Day
2. Q&A
3. CPU Scheduling
4. Wrap Up
Bug of the Day
void make_int_array(int* ptr, size_t count) {
    ptr = malloc(sizeof(int) * count);
}

int main() {
    int* arr;
    make_int_array(arr, 4);
    arr[0] = 0;
    arr[1] = 1;
    ...
}

**Explanation**
The make_int_array function changes ptr, but in that function ptr is essentially a local variable. That means the arr variable in main is unchanged.

**Bug Names**
Memory leak: ptr is malloced and never freed.
Uninitialized pointer write: arr is dereferenced but not initialized
Q&A
Assignment Questions

My code passes the autograder for the current directory, but fails all or nearly all tests when listing another directory. What gives?

When you open and list a directory (opendir and readdir) you get the names of entries in the directory. The names of entries in the current directory are valid paths. If you're listing a different directory, the names of files in that dir are not complete paths. You need to build a path to each file by concatenating a path to the directory with a "/" and the entry name.

Do we need to handle symbolic links?

No, just files and directories.
Lab Questions
CPU Scheduling
Scheduler Basics

Why do we need a CPU scheduler?

The CPU can't run everything at once, so we need to give each process time on the CPU.

If a program is going to run for a very long time, we don't want it to monopolize time on the CPU. A scheduler solves this by switching away to other jobs.

The OS virtualizes the CPU. The scheduler picks a process, then give it control of the CPU for some amount of time. Then we switch to another process.

The scheduler has to make decisions that result in "good performance"
**Scheduling Terminology**

**Batch Scheduler**
A scheduler that starts jobs and runs them to completion.

**Preemption**
When a scheduler interrupts a running job to switch to another job. This can happen in response to a variety of events: timers, new jobs, etc.

**Quantum**
A short unit of time that jobs can run before being preempted. Not all preemptive schedulers use a quantum, but some do.

**Context Switch**
When a scheduler switches from one job to another.
Scheduling Algorithms

First In, First Out (FIFO)
   Schedule jobs in the order they arrive.

Shortest Job First (SJF)*
   Schedule the shortest job first.

Shortest Time to Completion First (STCF)*
   Schedule the job that will finish first.

Round Robin
   Cycle through jobs executing each for one quantum.
Evaluating Schedulers

How can we measure a scheduler's performance? When is each measurement important?

Turnaround time: the time between when a job arrives and its completion.

Response time: the time between when a job arrives and its start.

Fairness: processes can't monopolize CPU time. Maybe CPU time is distributed evenly according to some formula.
Preemptive Scheduling

Most systems use preemptive schedulers. Why do you think this is the case? Without preemption, a process could run for a very long time uninterrupted. Other tasks would have to wait. With preemption, we can handle a long-running and short-running programs together. Users expect jobs to run fairly soon after input.

How do you decide between a long or short scheduler quantum?
Context switching takes some amount of time. The more we do it, the longer programs will take to finish. A long quantum decreases the rate of context switches. A short quantum will improve (reduce) response time.
Setting the quantum carefully allows us to amortize the cost of context switches.
Scheduler Examples
Illustrating Schedules

Show schedules for FIFO, SJF, STCF, and RR (quantum = 2).

A arrives T=0, length=4; B arrives T=1, length=2; C arrives T=3, length=1
FIFO

\[ RT = \frac{0 + 3 + 3}{3} = 2 \]
\[ TT = \frac{4 + 5 + 4}{3} = 4 \frac{2}{3} \]

SJF

\[ RT = \frac{0 + 4 + 1}{3} = 1 \frac{2}{3} \]
\[ TT = \frac{4 + 6 + 2}{3} = 4 \]
STCF

\[ RT = \frac{0+0+0}{3} = 0 \quad TT = \frac{7+2+1}{3} = 3 \frac{1}{3} \]

RR

\[ q = 2 \]

\[ RT = \frac{0+1+1}{3} = \frac{2}{3} \quad TT = \frac{7+3+2}{3} = 4 \]
Advanced Scheduling
(if there's time)
Advanced Scheduling *(if there's time)*

What kinds of workloads are particularly bad for these simple schedulers? Why?
For FIFO: if a long job arrives right before smaller jobs, the small jobs wait
For RR: two long jobs take longer if we context switch between them frequently
For STCF: jobs arriving in order of decreasing length leads the scheduler away from the long jobs and just runs the short ones.
For batch schedulers: many jobs arriving at once will be bad for response time

In general: workloads with a mix of interactive and long-running jobs do not fit well with any single scheduler.
Wrap Up
Reminders

Assignment
The listing directories assignment is due today

Lab
The memory allocator lab is due on Wednesday

Reading
Read the instructions for Wednesday's lab before class

Mentor Office Hours
Sunday 1–4pm
Tuesday 8–9pm
Wednesday 8–9pm