

Review

Computer Systems Programming, Spring 2024

Instructor: Travis McGaha

TAs:

Ash Fujiyama

Lang Qin

CV Kunjeti

Sean Chuang

Felix Sun

Serena Chen

Heyi Liu

Yuna Shao

Kevin Bernat

Logistics

- ❖ Project released
 - Due May 1st at midnight, please get started if you haven't already
 - Autograder posted now
 - NOTE: part of it is manually checked, not auto-graded

- ❖ HW4
 - Due last Friday
 - Autograder posted

- ❖ Last “Checkin” posted
 - Due May1st at midnight (late deadline over reading days)
 - (Post Semester Survey)

Today's Lecture

- ❖ Today's lecture is going to be bleh
 - I have some review materials prepared
 - There is other shit happening on campus
 - I know you all have projects and homework to wrap-up

- ❖ Poll:
 - Make this lecture Office Hours?
 - Next lecture (Wednesday) can still be review, and I am intending to have an exam review session during finals period.



Poll Everywhere

pollev.com/tqm

- ❖ Any questions? (On anything)
 - This is the chance for catchup questions, same at the beginning of next lecture.

Exam Philosophy / Advice (pt. 1)

- ❖ I do not like exams that ask you to memorize things
 - You will still have to memorize some critical things.
 - I will hint at some things, provide documentation or a summary of some things. (for example: I will list some of the functions that may be useful and a brief summary of what the function does)

- ❖ I am more interested in questions that ask you to:
 - Apply concepts to solve new problems
 - Analyze situations to see how concepts from lecture apply

- ❖ Will there be multiple choice?
 - If there is, you will still have to justify your choices

Exam Philosophy / Advice (pt. 2)

- ❖ I am still trying to keep the exam fair to you, you must remember some things
 - High level concepts or fundamentals. I do not expect you to remember every minute detail.
 - E.g. how a multi level page table works should be know, but not the exact details of what is in each page table entry
 - (I know this boundary is blurry, but hopefully this statement helps)

- ❖ I am NOT trying to “trick” you (like I sometimes do in poll everywhere questions)

Exam Philosophy / Advice (pt. 3)

- ❖ I am trying to make sure you have adequate time to stop and think about the questions.
 - You should still be wary of how much time you have
 - But also, remember that sometimes you can stop and take a deep breath.

- ❖ Remember that you can move on to another problem.

- ❖ Remember that you can still move on to the next part even if you haven't finished the current part

Exam Philosophy / Advice (pt. 4)

- ❖ On the exam you will have to explain things
- ❖ Your explanations should be more than just stating a topic name.
- ❖ Don't just say something like (for example) "because of threads" or just state some facts like "threads are parallel and lightweight processes".
- ❖ State how the topic(s) relate to the exam problem and answer the question being asked.

Review Topics

- ❖ Scheduling
- ❖ Threads
- ❖ IPC
- ❖ Networks (P1, P2, P3)
- ❖ Smart Pointers
- ❖ C++ Copying

NOTE: These are not all the topics that could be on the final. List is trimmed for review due to time constraints.

Will probably have a few more questions next lecture

In What order do the processes finish?

Scheduling

- ❖ The following processes are scheduled using a standard **Priority Round Robin** scheme.

Process Name	Arrival Time	Execution Time	Priority
Ape	0	7	medium
Bear	1	3	medium
Chinchilla	3	4	medium
Dolphin	4	4	low
Elephant	7	2	high
Flamingo	21	2	medium

- You may assume the following:

- the quantum for all processes (regardless of priority) is 2
- context switching is instantaneous
- if a process arrives and its priority is **higher** than that of the process that is currently running, the newly-arrived process is immediately scheduled; in that case, the process that is preempted goes to the end of its queue, but is able to run for a full quantum the next time it is scheduled
- if a process' time slice ends at the same time as another process of the same priority arrives, the one that just arrived goes into the queue **before** the one that just finished its time slice

Threads & Locks

- ❖ Consider we are working with a data base that has N numbered blocks. Multiple threads can access the data base and before they perform an operation, the thread first acquires the lock for the blocks it needs.
 - Example: Thread1 accesses B3, B5 and B1. Thread2 may want to access B3, B9, B6. Here is some example pseudo code:

```

void transaction(list<int> block_numbers) {
    for (every block_num in block_numbers) {
        acquire_lock(block_num)
    }

    operation(block_numbers);

    for (every block_num in block_numbers) {
        release_lock(block_num);
    }
}
    
```

Threads & Locks

- This code has the possibility to deadlock. Give an example of this happening. You can assume no thread tries to acquire the same lock twice
- Someone proposes we fix this by locking the whole database instead of locking at the block level. What downsides does this have? Does it even avoid deadlocks?

- How can we fix this (without locking the whole database if that even works)?

```
void transaction(list<int> block_numbers) {  
    for (every block_num in block_numbers) {  
        acquire_lock(block_num)  
    }  
  
    operation(block_numbers);  
  
    for (every block_num in block_numbers) {  
        release_lock(block_num);  
    }  
}
```

IPC

- ❖ The following code intends to use a global variable so that a child process reads a string and the parent prints it.
- ❖ Briefly describe two reasons why this program won't work. You can assume it compiles.

```

string message;

void child();
void parent();

int main() {
    pid_t pid = fork();
    if (pid == 0) {
        child();
    } else {
        parent();
    }
}

void child() {
    cin >> message;
}

void parent() {
    cout << message;
}
    
```

IPC

- ❖ Describe how we would have to rewrite the code if we wanted it to work. Keeping the multiple processes and calls to `fork()`. Be specific about where you would add the new lines of code.

```
string message;

void child();
void parent();

int main() {
    pid_t pid = fork();
    if (pid == 0) {
        child();
    } else {
        parent();
    }
}

void child() {
    cin >> message;
}

void parent() {
    cout << message;
}
```

Networking: pt. 1

- ❖ TCP guarantees reliable delivery of the packets that make up a stream, assuming that the socket doesn't fail because of an I/O error.
- ❖ IP guarantees reliable delivery of packets, assuming that the socket doesn't fail because of an I/O error.
- ❖ Given a particular hostname (like `www.amazon.com`), `getaddrinfo()` will return a single IP address corresponding to that name.
- ❖ A single server machine can handle connection requests sent to multiple IP addresses.
- ❖ A struct `sockaddr_in6` contains only an ipv6 address.
- ❖ The HTTP payload takes up a larger percentage of the overall packet sent over the network than the IP payload.

Networking pt. 2

- ❖ For each of the following behaviors, identify what networking layer is most closely thought of as being responsible for handling that behavior.
 - Host A tries to send a long message to Host B in another city, broken up into many packets. A packet in the middle does not arrive, so Host A sends it again.
 - Host A tries to send a message to Host B, but Host C and Host D are also trying to communicate on the same network, so Host A must avoid interfering

Networking pt. 3

- ❖ The original versions of HTTP (including 1.1) were designed to use plain text characters sent over the network instead of alternatives like a binary encoding for the request and response. Describe one advantage of this design decision and one disadvantage.
- ❖ Advantage:
- ❖ Disadvantage:

Smart Pointers

- ❖ Suppose we have the following declarations at the beginning of a C++ program:

```
int n = 17;  
int *x = &n;  
int *y = new int(42);
```

- ❖ For each part, indicate whether if we were to add just that line(s) after the code above, whether there is a compiler error, some sort of run time error, or memory leak.
 - `unique_ptr a(n);`
 - `unique_ptr b(x);`
 - `unique_ptr c(y);`
 - `unique_ptr d(&n);`
 - `unique_ptr e(new int(333));`
 - `unique_ptr temp(new int(0));`
`unique_ptr f(temp.get());`

C++ Copying

- ❖ Below is a class that represents a Multiple Choice answer

```
class MC {
public:
    MC() : resp_(' ') { }
    MC(char resp) : resp_(resp) { }
    char get_resp() const { return resp_; }
    bool Compare(MC mc) const;
private:
    char resp_;
}; // class MC
```

- ❖ How many times are each of the following invoked:
 - MC constructor
 - MC copy constructor
 - MC operator=
 - MC destructor

```
int QS = 2
// this works
MC key[2] = {'D', 'A'};

size_t Score(const MC *ans) {
    size_t score = 0;
    for (int i = 0; i < QS; i++) {
        if (ans->Compare(key[i])) {
            score++;
        }
        ans++;
    }
    return score;
}

int main(int argc, char **argv) {
    MC myAns[QS];
    myAns[0] = MC('B');
    myAns[1] = MC('A');
    cout << "Score: ";
    cout << Score(myAns) << endl;
    return 0;
}
```