Problem Set 4: Deadlock & Main Memory1

- See course webpage for due date and time.
- Submit deliverables to CourSys: https://courses.cs.sfu.ca/
- Late penalty is 10% per calendar day (each 0 to 24 hour period past due).
  - Maximum 2 days late (20%)
- Do not show another student your code, do not copy work found online, and do not post questions about the assignment online. Please direct all questions to the instructor or TA: cmpt-300-d2-help@sfu.ca;
  You may use general ideas you find online and from others, but your solution must be your own.
- See the marking guide for details on how each part will be marked.

When asked to explain something, don't copy and paste text found online or in the man pages. Explain the idea in your own words.

Short Answer Questions

- Write brief answers to the following questions in a text file name `short_answer.txt` or write your answers in another program and generate a PDF named short_answers.pdf.

Chapter 7 - Deadlock2

1. Using a resource allocation graph, create an example of a system in an unsafe state such that:
   - It uses at least three processes.
   - It is possible for it to enter deadlock.
   - It is possible for it to not enter deadlock.

   a. Show or briefly explain how the system can deadlock.

   b. Show or briefly explain how all the processes could complete without entering deadlock.

   c. Explain how the CPU scheduler can be a factor in determining if a system deadlocks or not.

2. Figure 1: Possible traffic deadlock pattern.

   a. Show that the four necessary conditions for deadlock hold in this example.

   b. State a simple rule for avoiding deadlock in this system.

3.

1. Consider the following snapshot of a system:

	Allocation	**Max**	**Available**
	A B C D	**A B C D**	A B C D
P_0	2 0 0 1	4 2 1 2	3 3 2 1

P_1	3 1 2 1	5 2 5 2	
P_2	2 1 0 3	2 3 1 6	
P_3	1 3 1 2	1 4 2 4	
P_4	1 4 3 2	3 6 6 5	

1.

Answer the following questions using the banker's algorithm:

a. Show that the system is in a safe state by determining an order in which processes may complete.

b. If a request from process P_1 arrives for $(1, 1, 0, 0)$, can the request be granted immediately? Show your work using the banker's algorithm as needed.

c. If a request from process P_4 arrives for $(0, 0, 2, 0)$, can the request be granted immediately? Assume it is working from the initial state (i.e., that P_1's request did not occur). Show your work using the banker's algorithm as needed.

d. If a request from process P_3 arrives for $(0, 0, 0, 3)$, can the request be granted immediately? Assume it is working from the initial state (i.e., that P_1's and P_4's requests did not occur). Show your work using the banker's algorithm as needed.

Chapter 8: Main Memory1

1. Why are page sizes always powers of 2?
2. Explain the difference between internal and external fragmentation.
3. Compare the memory organization schemes of contiguous memory allocation, pure segmentation, and pure paging with respect to the following issues:
   a. External fragmentation
   b. Internal fragmentation
   c. Ability to share code across processes
4. What is the purpose of paging the page table?

Working in the Linux Kernel

- Follow the Custom Kernel Guide posted on the course website to download the Linux kernel source code, compile it, and start it running in a system. It is suggested you use the QEMU virtual machine, but you may use any other method of executing the kernel you like.

Compile Fibonacci Code for VM

- Copy the Fibonacci code you wrote for Problem Set 2 (or you may use the

sample solution, for no penalty) if you did not successfully write the Fibonacci code) into a new folder for problem set 4.

- Create a Makefile to build the code using the `-static` GCC build flag. Have the `make all` command build an executable named `fibonacci`
- Add a `clean` target to the Makefile to delete the `fibonacci` output file.
- Add a `transfer` target to the Makefile which executes the SCP command to copy the `fibonacci` executable into the QEMU system. (See Custom Kernel Guide for SCP details).
    - Hint, you can execute any commands in a Makefile. For example, to have a target named `test` which echos a message, one could use the following code in the Makefile:
      ```
      test:
            echo Hello world!
      ```
- Add a killqemu target to the Makefile which executes the killall command listed in the Custom Kernel Guide for killing all open QEMU virtual machines.

VM Interaction

- Ensure that you have compiled the Linux kernel with the "Local version" string set to be your SFU ID.
- Start the VM running your custom kernel and then capture the text (via copy-and-paste is fine) of the terminal session where you run the below commands. Copy and paste the text into a file name qemu_capture.txt

1. Log in as `root`.
2. SCP fibonacci from the host to the guest OS (inside QEMU). This will not generate any output in QEMU.
3. `# ls -la`
4. `# uname -a`
5. `# cat /proc/uptime`
6. `# cat /proc/version`
7. `# ./fibonacci 10`

- The output you capture should show the log-in prompt (from "Debian GNU/ Linux 6.0 …" before typing in the user name) until you finish the last command. If you make an error, don't worry about it, just keep capture the text.

Deliverables

Submit the following deliverabels in a ZIP or tar.gz file to CourSys:

1. `short_answer.txt` (or .pdf)
2. `fibonacci.c`
3. `Makefile`
4. `qemu_capture.txt`

- Please remember that all submissions will automatically be compared for unexplainable similarities.