Memory and Context Switching

ECE598: Advanced Operating Systems – Homework 6

Spring 2018

Due: Thursday, 22 March 2018, 2:00pm

This homework involves memory allocation and context switching.
You make work in a group on this assignment.

1. **Download the homework code template**

   - Download the code from:
     `http://web.eece.maine.edu/~vweaver/classes/ece598/ece598_hw6_code.tar.gz`
   - Uncompress the code. On Linux or Mac you can just
     `tar -xzvf ece598_hw6_code.tar.gz`
   - Note! The code layout has changed drastically since the previous homework!
     - Things have been moved to subdirectories
     - The userspace code is built separately and put into a ROMFS disk image.
     - Try building and running the code as soon as possible! I've only really tested on Linux so if it breaks for you let me know so I can address this.

2. **Context switch**

   - The code has been updated to allow full multi-tasking!
   - First, `kernel/drivers/timer/sp804_timer.c` was updated so the interrupt happens at 64Hz.
   - `vfork()` and `execve()` were implemented to allow starting new processes. See the code in `kernel/processes/`
   - A scheduler was added `kernel/processes/scheduler.c` that is run on every timer interrupt.
   - When no jobs are ready, an idle task that does nothing is run.
   - Test it out! Run `printa &` (the ampersand means run in the background). Then quickly type `printb` and you should see both jobs are running at the same time.
   - Now go and answer question 4a below.

3. **Memory allocation code (4pts)**

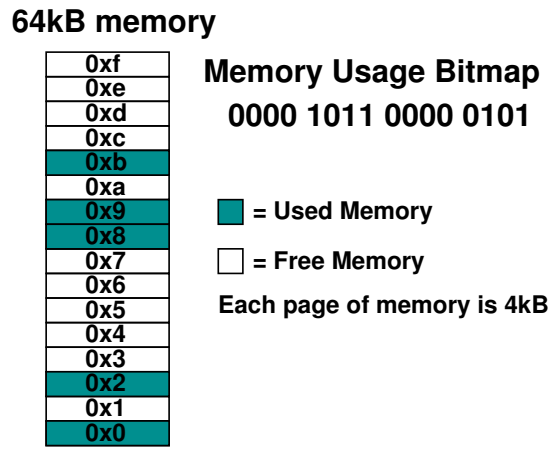
   - Look at the memory allocation code in the `find_free()` function in `kernel/memory/memory.c`.
   - Determine what type of algorithm is used, and answer Question 4b below.
   - Modify the `memory.c` file so it implements the next-fit algorithm.

4. **Questions (6pts)**

   Answer these questions in the README file.

   (a) What type of scheduling algorithm is implemented in `kernel/processor/scheduler.c`?

   (b) What type of memory allocation algorithm is implemented in `kernel/memory/memory.c`?



   (c) In the above diagram, how much memory is free?

   (d) If you were allocating a 16kB chunk of memory using the first-fit algorithm, where would it go?

   (e) If you were allocating a 16kB chunk of memory using the best-fit algorithm, where would it go?

   (f) In this case, why might the best fit result be better than the first fit one?

   (g) Would it be possible to allocate a 32kB chunk of RAM?
   If not, what could be done to make this possible?
   Would the proposed action work if the chunks of memory shown were allocated by a C program using malloc()?

5. **Submit your work**

   - Run `make submit` in your code directory and it should make a file called `hw6_submit.tar.gz`. E-mail that file to me as well as the document with the answers to the questions.