Programming Assignment 1

Start Assignment

- Due Saturday by 11:59pm
- Points 12
- Submitting a website url
- Available Aug 13 at 12am - Aug 23 at 11:59pm

Introduction

For the programming assignment 1, we will look into the xv6 (https://canvas.instructure.com/courses/12589532/files/307841777?wrap=1) (https://canvas.instructure.com/courses/12589532/files/307841777/download?download_frd=1) operating system and add some functionalities there.

1. We will need some tools/frameworks that will help us work with the xv6 operating system:
    1. **ARM GCC Toolchain**: `arm-none-eabi-gcc`
    2. QEMU: For ARM system emulation and testing.
    3. GNU-MAKE: Build utility
2. Once these are installed and you have downloaded the code tarball, run the following to see the OS in action:

    1.
       ```
       prompt> make clean
       ```

    2.
       ```
       prompt> make qemu
       ```
3. If everything compiles correctly, you should see the following:

    ```
    xv6 kernel is booting

    hart 2 starting
    hart 1 starting
    init: starting sh
    $
    ```

4. Run ls to see a list of commands/user programs available in the starter OS.
5. Take some time to understand the code in the different files in the folder. Below are some description for important files:
    1. user.h contains the system call definitions in xv6.
    2. usys.S contains a list of system calls exported by the kernel, and the corresponding invocation of the trap instruction.
    3. syscall.h contains a mapping from system call name to system call number. Every system call must have a number assigned here.
    4. syscall.c contains helper functions to parse system call arguments, and pointers to the actual system call implementations.
    5. sysproc.c contains the implementations of process related system calls.

6. defs.h is a header file with function definitions in the kernel.
7. proc.h contains the struct proc structure.
8. proc.c contains implementations of various process related system calls, and the scheduler function. This file also contains the definition of ptable, so any code that must traverse the process list in xv6 must be written here.

Things to do

Implement the following to better understand how the system runs.

1. (2 pts.) "uptime" command to print how long has the system been running
2. (3 pts.) pause <no. of seconds>
    1. You can put your code in user/pause.c. Look at some of the other programs in user/ (e.g., user/echo.c, user/grep.c, and user/rm.c) to see how command-line arguments are passed to a program.
    2. Add your pause program to UPROGS in Makefile; once you've done that, make qemu will compile your program and you'll be able to run it from the xv6 shell.
    3. If the user forgets to pass an argument, sleep should print an error message.
    4. The command-line argument is passed as a string; you can convert it to an integer using atoi (see user/ulib.c).
    5. Use the system call sleep to enforce this program.
    6. See kernel/sysproc.c for the xv6 kernel code that implements the sleep system call (look for sys_sleep), user/user.h for the C definition of sleep callable from a user program, and user/usys.S for the assembler code that jumps from user code into the kernel for sleep.
    7. pause's main should call exit(0) when it is done.
3. (3 pts.) modify the shell to support tab completion, i.e., pressing the tab key should try to complete the command if it is unique and list the possible commands if they are not unique as per the string matching upto the text provided by the user
4. (4 pts.) ps command that prints details of all processes running on the system
    1. You may have to implement various system calls as part of this command
    2. You should print the process-id, parent's process id, name of the process, state of the process, and an additional field which prints the number of system calls the process has invoked.

Submission

- Join the GitHub classroom here to create a repository: https://classroom.github.com/a/h2nLSOuW (https://classroom.github.com/a/h2nLSOuW)
- Download the starter code (link provided above) and upload that to the repo after you check that it is working fine on your system.
- You can then upload your code-changes to the same repo and any changes that you make through commits (More commits indicate better approach and less chances of plagiarism!)
- Share the repo link as the submission for this assignment on Canvas to complete the assignment.