UP24 Lab02 (Pre-Lab Announcement)

Date: 2024-02-27

Implement a Kernel Module

Our next lab is to practice implementing a kernel module. The details will be announced next week. Before you play with our lab next week, here are some hints for you to prepare before our lab officially announced.

Preparation

You may not have experience in implementing a kernel module. Before you start your implementation, you may read some relevant kernel documents and tutorials.

- Please check the `file+stdio` course slide and read the `hellomod` example to see how a simple kernel module is implemented.
- The Linux kernel documentation (https://www.kernel.org/doc/html/latest/), including
  - ioctl based interface (https://www.kernel.org/doc/html/latest/driver-api/ioctl.html)
  - Memory allocation guide (https://www.kernel.org/doc/html/latest/core-api/memory-allocation.html)
  - Memory management APIs (https://www.kernel.org/doc/html/latest/core-api/mm-api.html)

Our development package (including runtime and development files) can be found here (dist-6.6.17.tbz) (https://up.zoolab.org/unixprog/lab02/dist-6.6.17.tbz). You may download and play with it before our lab officially starts. Updated `hellomod` codes is also provided here (hellomod-6.6.17.tbz) (https://up.zoolab.org/unixprog/lab02/hellomod-6.6.17.tbz) for you.

The runtime contains a pre-built Linux kernel, a root filesystem, the modules required to build a module, and a script to boot the system with the QEMU emulator. To boot the sysetem, unpack the `dist` tarball and run the `qemu.sh` command.

You can develop the module on Apple chip macs but note that all the files must be cross-compiled to x86_64 architecture. We have created another docker-based runtime for you to build a cross-compilation environment for UNIX-based environment (Mac OS or WSL). You may download the files from here (crossbuild.tbz) (https://up.zoolab.org/unixprog/lab02/

crossbuild.tbz). To run the runtime, please follow the steps below.

1. Unpack `crossbuild.tbz` , a `corssbuild` directory will be created with a build script and `Dockerfile` .
2. run `build.sh` in the `crossbuild` directory, a docker images called `chuang/build` will be created.
3. To compile your codes, ensure that you have the environment variables `UID` and `GID` setting to the user id and group id of the current user. Then, switch to your working directory and run the command:

   ```
   docker run -it --rm --user "$UID:$GID" -v "`pwd`:/build" -w /build -e PS1
   ```

   You can then cross-compile x86-64 binaries using the compiler `x86_64-linux-gnu-gcc` to compile and generate binaries running on x86-64 platform.

   > The cross-compilation command for building x86 modules on arm64 is `make ARCH=x86 CROSS_COMPILE=x86_64-linux-gnu-`

> Our course video `file+stdio` has introduced how `ioctl` works with a kernel module. This lab extends it by implementing more features in the kernel module.