Tools Used in CPSC422/522

If you use a Zoo machine that runs Linux, then most of the software needed will be installed locally. To set the correct paths run `source /c/cs422/env.sh` .

If you're using a personal machine running Debian/Ubuntu, you can install the necessary tools by running:

```
1.   sudo apt-get install -y build-essential gcc-multilib gdb
```

We highly recommend using Linux or Zoo machines. However, if you have a Mac OS X machine, you can compile the toolchain, which should allow you to do the labs. Do not attempt to work on the labs while using Windows.

For an overview of useful commands in the tools used in CPSC422/522, see the lab tools guide (/cs422/labguide).

Compiler Toolchain

> Note: The course staff have not verified that these instructions still work. They are reproduced from MIT's 6.828 (http://pdos.csail.mit.edu/6.828/2012/).

Most modern Linuxes and BSDs have an ELF toolchain compatible with the labs. That is, the system-standard `gcc` , `as` , `ld` and `objdump` should just work. The lab makefile should automatically detect this. However, if your machine is in this camp and the makefile fails to detect this, you can override it by adding the following line to `conf/env.mk` :

```
1.   GCCPREFIX=
```

If you are using something other than standard x86 Linux or BSD, you will need the GNU C compiler toolchain, configured and built as a cross-compiler for the target `i386-jos-elf` , as well as the GNU debugger, configured for the `i386-jos-elf` toolchain. You can download the specific versions we used via these links, although any recent versions of gcc, binutils, and GDB should work:

- http://ftpmirror.gnu.org/binutils/binutils-2.21.1.tar.bz2 (http://ftpmirror.gnu.org/binutils/binutils-2.21.1.tar.bz2)
- http://ftpmirror.gnu.org/gcc/gcc-4.5.1/gcc-core-4.5.1.tar.bz2 (http://ftpmirror.gnu.org/gcc/gcc-4.5.1/gcc-core-4.5.1.tar.bz2)
- http://ftpmirror.gnu.org/gdb/gdb-6.8a.tar.gz (http://ftpmirror.gnu.org/gdb/gdb-6.8a.tar.gz)

Once you've unpacked these archives, run the following commands as root:

```
$ cd binutils-2.21.1
$ ./configure --target=i386-jos-elf --disable-werror --disable-nls
$ make
$ make install
$ cd ../gcc-4.5.1
$ ./configure --target=i386-jos-elf --disable-nls --without-headers \
              --with-newlib --disable-threads --disable-shared \
              --disable-libmudflap --disable-libssp
$ make
$ make install
$ cd ../gdb-6.8
$ ./configure --target=i386-jos-elf --program-prefix=i386-jos-elf- \
              --disable-werror
$ make
$ make install
```

Then you'll have in /usr/local/bin binaries with names like i386-jos-elf-gcc. The lab makefile should detect this toolchain and use it in preference to your machine's default toolchain. If this doesn't work, there are instructions on how to override the toolchain inside the GNUmakefile in the labs.

QEMU Emulator

QEMU (http://www.nongnu.org/qemu/) is a modern and fast PC emulator. A special version of QEMU version 1.7.0, as described below, is set up on the Zoo. Add it to your path by executing `source /c/cs422/env.sh`.

Unfortunately, QEMU's debugging facilities, while powerful, are somewhat immature, so we highly recommend you use our patched version of QEMU instead of the stock version that may come with your distribution. The version installed on the Zoo is already patched. To build your own patched version of QEMU:

1. Clone the our QEMU git repository
   `git clone https://github.com/davidiw/qemu.git -b 6.828-1.7.0`
2. On Linux, you may need to install the SDL development libraries to get a graphical VGA window. On Debian/Ubuntu, this is the `libsdl1.2-dev` package. Furthermore, the Pixman module (http://www.pixman.org/) is needed to build QEMU. On Debian/Ubuntu, this is the `libpixman-1-dev` package.
3. Configure the source code
   1. Linux:
      `./configure --disable-kvm [--prefix=PFX] [--target-list="i386-softmmu x86_64-softmmu"]`
   2. OS X:
      `./configure --disable-kvm --disable-sdl [--prefix=PFX] [--target-list="i386-softmmu x86_64-softmmu"]`
      The `prefix` argument specifies where to install QEMU; without it QEMU will install to `/usr/local` by default. For example, a useful prefix that will not require root access is `$HOME/apps/qemu`. The `target-list` argument slims down the architectures QEMU will build support for.
4. Run `make && make install`