CS 35L Software Construction Laboratory (Lab3-A)
Mon, Jan 23, 2012, Ver 1.1

Tarball

```
  What is a tarball? -- commonly used to refer to a file which contains other
                        files. Tar program itself does not compress the files.
                        Actually, tar works with a compression program like gzip
                        to compress the file
  An example:
  ftp://ftp.gnu.org/gnu/coreutils/coreutils-7.6.tar.gz
  .tar extension is for the actual tarball
  .gz extension suggests that this tarball is compressed by gzip
  .tar.gz is equivalent to .tgz
  A quick extract command:
  tar vxfz coreutils-7.6.tar.gz
    v -- produce verbose output. (optional)
    x -- extract files from an archive.
    f -- read the archive from or write the archive to the specified file.
    z -- compress the resulting archive with gzip.
         in extract or list modes, this option is ignored.
  A quick compress command:
  tar vcfz coreutils-7.6.tar.gz coreutils-7.6
    Note: coreutils-7.6.tar.gz is the tarball file which will be created
          coreutils-7.6 is the directory which will be packed and compressed
  More reading:
  http://maketecheasier.com/install-software-from-a-tarball-in-linux/2009/06/25
```

Basic gcc/g++

```
  gcc is used for c and g++ is used for c++

    +-------------+  compile   +--------------+  link   +----------------+
    | source code | ---------> | object files | ------> | target program |
    +-------------+            +--------------+         +----------------+

  -- one (.c) source file will generate one object file
     g++ -o kernel.o -c kernel.cc
     g++ -o gui.o -c gui.cc
  -- link (combine) multiple object files into one target program
     g++ -o program kernel.o gui.o
```

configure and make

```
    When you get source code from others, you can try the following set of commands
  to "install" that program:
    ./configure
    make
    sudo make install

    Note: -- "configure" file is a executable script which automatically generates
             Makefile
          -- "Makefile" contains a set of rules which specify how to derive the
             target file
```

```
        -- When you run command "make", try "make -j2". With "-j2", the make
           program will create two parallel threads to speed up the compilation

    +--------------+                    +------+
    | ./ configure | ---> Makefile ---> | make |
    +--------------+                    +------+

  A sample make file:
   +----------------------------------------------+
    CC     = gcc
    CFLAGS = -g

    all: helloworld

    helloworld: helloworld.o
            # Commands start with TAB not spaces
            $(CC) $(LDFLAGS) -o $@ $<

    helloworld.o: helloworld.c
            $(CC) $(CFLAGS) -c -o $@ $<

    clean:
            rm -f helloworld helloworld.o
   +----------------------------------------------+
  Notes:
    Special macro: $<   dependant file
                   $@   target file
  More reading:
  -- Make (wiki page)
    http://en.wikipedia.org/wiki/Make_(software)
  -- a tutorial of using autoconf and automake with C++:
    http://www.openismus.com/documents/linux/automake/automake.shtml

New commands that you need to know:
  1. tar 2. gcc 3. g++ 4. make

Preview Python: http://docs.python.org/release/2.4.1/tut/tut.html
```