1) The gzip utility:

i.

- What URL can you download it from?
  * http://ftp.gnu.org/gnu/gzip/ (or local mirror ftp://ftp.heanet.ie/pub/gnu/gzip/)
- What's the latest version?
  * The latest version is 1.8.
  * Confusingly the project home page (http://www.gzip.org/) talks about 1.2.4 and a beta 1.3.3, but that info is well out of date.
- What version is currently on chuck?
  * 1.5 currently (found with 'gzip --version' or 'rpm -q gzip')
- What would be an advantage of downloading and installing it yourself?
  * The system/packaged version on chuck is a number of versions behind. If there are any new features or bugfixes in the more recent versions, it's a good idea to use the latest.
  * There could be other pieces of software which depend on a newer version of gzip, so you may have to install it as a dependency of something else.

ii.

- List a couple of differences in the log of changes (hint) between the system version and the one you've downloaded.
  * Typically the 'ChangeLog' file is used to write a record of major changes between versions.
  * Some changes: "fix bug with -l output to pipes"; "also honor GZIP=--rsyncable"; some notes about porting to Solaris and NetBSD.

iii.

- How do you find out how to compile it - which file(s) should you read?
  * Typically there is a 'README' file describing the software and potentially containing notes about how to compile.
  * More usually there is a separate 'INSTALL' file (the README should point you there if so) with instructions on how to compile and install.
- Why in CAPS?
  * The filenames are in CAPS to make them stand out. In particular, traditionally in UNIX environments with ASCII ordering, 'ls' will list files with UPPERCASE first, followed by lowercase; so the INSTALL and README files would usually be at the top of the 'ls' listing (note that many UNIX/Linux environments do not currently sort files in this way).

iv.

- What happens if you just type 'make' and skip the first step?
  * With GNU software (specifically the autoconf/autotools) build system, you must typically run './configure' first before it will create a Makefile. I.e. there is no Makefile to begin with. There is usually a 'Makefile.in' template file, and the './configure' step creates a real Makefile using that as the template and filling in the blanks with information gleaned from the local system. It configures the generic 'Makefile.in' with the specific details of the OS and environment in question.
- Why do we need the './' in the first step?
  * The current directory is typically not in your PATH, so trying to run 'configure' will not work; we need './configure' to explicitly use the PATH to the script.

v.

- What are the typical configuration, compilation and installation steps?
  * The autoconf/autotools build system has these three steps: './configure; make; make install'.
  * The first step './configure' is used to customise the build process for the current system. Additional flags may be passed to ./configure to manually specify values or change defaults.
  * The second step 'make' compiles the software (this uses the default make target in the Makefile).
  * Optionally there is usually a 'make check' or 'make test' test target, which runs a test-suite for the software.
  * The third step 'make install' (using the extra 'install' target in the Makefile) installs the software into the designated prefix ('/usr/local' by default).

vi.

- Where would it try to install by default?
  * By default it expects to install with a prefix of '/usr/local' (so executables will go into '/usr/local/bin', config files will go into '/usr/local/etc', and so on). A different installation prefix can be passed to the configure script, for example './configure --prefix=$HOME/software'
- Why won't that work for you on chuck?
  * The default won't work for you on chuck, because you need admin/root rights to be able to copy files into '/usr/local'. On your home system, you could do this with 'sudo' or 'su' or similar.
- How do you change the installation prefix to somewhere in your homedir?

```
        * Pass an argument to the configure script: './configure --prefix=$HOME/software'
      - Why is setting the prefix to just $HOME a bad idea; what prefix did you use?
        * Just using './configure --prefix=$HOME' will lead to a very messy and cluttered HOME
directory. You don't know ahead of time what files/directories 'make install' will put into the
destination directory, and this could easily lead to conflicts. It is much better practice and safer
to explicitly have a folder '$HOME/software' or '$HOME/build', so that you know exactly what you have
put in there.
   vii.
      - How long does the compilation step take?
        * For me, it took 4.3 seconds (note the easiest way to measure this is to prefix the command
with the 'time' command; that is, run 'time make').
      - If you do a 'make clean' followed by 'make -j 2', how does that impact on the time? Why?
        * Note that if you went straight to 'time make -j 2' then there is nothing to do, as the
'make' has already finished; that's why we need to run 'make clean' first.
        * For me, 'time make -j 2' took 2.2 seconds.
        * The '-j 2' flag tells 'make' to run two compilation jobs in parallel, using 2 CPU cores to
build in parallel, if possible. Hence, depending on how the Makefile is structured with dependences,
and how many files need to be compiled, we could expect a speed-up of roughly 2 in the build process.
In practice, it's unlikely to be exactly 2 due to things like the Makefile dependency tree and the
cost of building it, and the final linking steps are not parallel usually; also external factors (is
someone else building on chuck at the same time?). Note also that if there is only a single CPU core
on the machine, then 'make -j 2' will take roughly the same time as 'make'.
   viii.
      - Are there any self-test targets available? Does it pass?
        * There is a 'make check' self-test target. For me, 18 tests passed out of 18.

2) The wget utility:
   i.
      - What URL can you download it from?
        * http://ftp.gnu.org/gnu/wget/ (or local mirror ftp://ftp.heanet.ie/pub/gnu/wget/)
      - What's the latest version?
        * 1.19 currently
      - What version is currently on chuck?
        * 1.14 currently (found with either 'wget --version' or 'rpm -q wget').
   ii.
      - What are the compilation steps?
        * The same as gzip: './configure; make; make install'
   iii.
      - Why does the first step fail?
        * It fails because it cannot find a TLS/SSL library. These are the error messages: "checking
for GNUTLS... no; configure: error: Package requirements (gnutls) were not met: No package 'gnutls'
found"
      - How would you fix it?
        * Try './configure --help' to see what options are available, in particular relating to
'gnutls'. Searching for 'gnutls' in the help output we see that it is looking for an SSL library '--
with-ssl' and while it defaults to 'gnutls', it will also work with 'openssl'.
        * Using that knowledge, re-run the configure script and tell it to use openssl instead:
'./configure --with-ssl=openssl'
        * Note: alternatively you could download and build gnutls, and point the wget configure
script to wherever you installed it. I note that I did not attempt this on chuck.
   iv.
      - Does it pass the self-tests?
        * For me, 83 tests passed out of 85, with 2 skipped.
      - What command did you use?
        * There is a 'make check' self-test target.
   v.
      - How long does the compilation step take?
        * 7.5 seconds for me.
      - If you do a 'make clean' followed by 'make -j 2', how does that impact on the time? Why?
        * It now took 3.4 seconds for me. Same as for 'gzip', '-j2' will make the compilation step
run in parallel over 2 CPU cores. It is very roughly twice as fast.
   vi.
      - What prefix did you set?
        * './configure --prefix=$HOME/software --with-ssl=openssl'
      - How would you make sure that you use the custom wget utility instead of the system one?
```

```
        * You could add this line to your .bashrc to change your PATH: 'export
PATH=$HOME/software/bin;$PATH'

3) The openssl encryption suite:
   i.
      - What URL can you download it from?
        * https://www.openssl.org/source/ (or local mirror
ftp://ftp.heanet.ie/pub/ftp.openssl.org/source/)
      - What's the latest version in the 1.0.1 branch?
        * 1.0.1u currently
        * Note that 1.0.1 has been fully deprecated, so can now be found in the 'old' source url:
https://www.openssl.org/source/old/1.0.1/
      - What version is currently on chuck?
        * 1.0.2k currently (found with either 'openssl version' or 'rpm -q openssl')
   ii.
      - What are the compilation steps?
        * './config; make; make test; make install'
      - Are there any differences to the GNU software?
        * It doesn't use the autoconf/autotools build system; it uses its own configure (config)
script. There is a Makefile by default, but the software still won't build without the './config'
step first as the Makefiles are incomplete.
   iii.
      - How long does the compilation step take?
        * For me, 1 minute 19 seconds.
      - If you do a 'make clean' followed by 'make -j 2', does that impact on the time? Why not?
(hint: any warnings at the top of the make output?)
        * 1 minute 8 seconds; no real impact on compilation time.
        * There is a very cryptic warning at the top of the make output "make[1]: warning: jobserver
unavailable: using -j1.  Add `+' to parent make rule". Deciphering this message (using a search tool
of your choice) indicates that it means that the Makefile is not structured in a way that allows
parallel builds; it cannot create a parallel build tree.
   iv.
      - What is the latest version?
        * 1.1.0g currently
      - Does 'make -j 2' make a difference to compiling for this version?
        * It speeds up from 1 minute 48 seconds to 54 seconds. It seems that the latest version has
fixed the Makefile issue, allowing parallel builds.
```