

Basic Linux Recap

- How do you open a console window?
- What do you type to open pico? how about gedit?
- When you see the '^' symbol you should press what key? (pico)
- What are the keyboard shortcuts for save and quit in pico?
- How do you copy/paste in pico?
- How can you keep from typing 80 characters per line in pico?
- How do you enable syntax highlighting in gedit?
- How can you open gedit without hanging your terminal?
- How can gedit help keep you from typing more than 80 characters on a line?
- Browse to your home directory and:
 - Create a directory called CS270 (mkdir)
 - Jump to the CS270 directory (cd)
 - Create a directory called PA0 (mkdir)
 - Jump to the PA0 directory (cd)
 - Create a file called README (pico/gedit)
 - In the README file type your full name and your username.
 - Save the file

SVN

- What is the difference between a repository and a working directory?
- Should you edit files in the repository?
- What do you gain from using SVN?
- Explain what the following SVN commands do:
 - svn checkout
 - svn add
 - svn commit
 - svn update
 - svn diff
 - svn log
 - svn info
- If working on code with a group, how often should you run svn update?
- How can you view the files in an SVN repository?
- Browse to your home directory and (use svn notes from class website):
 - Create an SVN repository for PA0 (notes).
 - Import your existing PA0 code into the repository (notes).
 - Checkout the repository into your existing working directory (notes).

Make

- What should a makefile be called?
- What is a makefile?
- Browse to your working directory for PA0 and:
 - Copy the makefile from the class directory to this directory and rename it (cp)
 - Attempt to build your makefile (you should get an error)
- Put the makefile into the SVN repository
 - Add the makefile into the repository (svn add)
 - Commit your changes into the repository (svn commit)
 - You may get an error regarding messages, how can you fix this?

GDB

- Explain what each of the following gdb commands does:
 - run
 - cont
 - step
 - next
 - print
 - list
 - where
- Name two ways to set a breakpoint in gdb:
- Browse to your cs270 directory (~username/cs270) and:
 - Create a directory named R0 (mkdir)
 - Browse to the R0 directory (cd)
 - Copy the contents of ~cs270/public/R0 into this directory (cp)
 - Build the makefile (make)
 - Run the program Example1
- You should have encountered a segmentation fault, we can debug this with gdb.
 - Run gdb on the program, then run your program (no breakpoint)
 - Why no breakpoint?
 - Use gdb to figure out the following
 - What line does it crash on?
 - What data type is 'result'?
- Fix the program and re-make it. Now the multiplyTwoNumbers function is still behaving incorrectly (if it's working for you, try a different input value!). We can use gdb to find out why.
 - Run gdb on the program.
 - Set a breakpoint at the beginning of the multiplyTwoNumbers function.
 - Run your program.
 - Figure out what's wrong.

C

- In a C program the _____ files (.___) contain the declaration and the implementation is contained in the _____ files (.___)
- What is the name of the standard I/O library in C?
- What is a pointer used for?
- What is a NULL pointer?

- Observe the attached code. Assume the user executes `exampleFunction()`. When the code reaches `“//Some Comment”`:
 - What is the value of `val.x`?
 - What is the value of `ref.x`?
 - What is the value of `ref.y`?
 - What is the `(->)` operator used for?
 - How could we replace the `(->)` operator?
 - Does this do anything different?
 - What’s the difference between a string and an array of characters?
 - What does the command `printf` do?
 - How many arguments does `printf` take?

- Explain the following `printf` flags
 - `%d`
 - `%i`
 - `%n`
 - `%f`
 - `%c`
 - `%s`

- Observe the following `printf` statements

```
int number = 5;  
char str[20] = “hello”;  
double fraction = 0.123;
```

1. `printf(“%s world”, str);`
2. `printf(“My favorite fraction is %d”, fraction);`
3. `printf(“I don’t like arguments”);`

- Which statement(s) probably don’t display what was intended?
- What is a segmentation fault?

Foo.h

```
struct parameterBlock {  
  
    int x;  
    unsigned int y;  
  
};  
void byReference(parameterBlock block);  
void byValue(parameterBlock * block);  
void exampleFunction(void);
```

Foo.c

```
void byReference(parameterBlock * block)  
{  
    block.x = 5;  
}  
void byValue(parameterBlock block)  
{  
    block->x = 5;  
}  
void exampleFunction(void)  
{  
    parameterBlock val;  
    val.x = 6;  
    val.y = 7;  
  
    byValue(val);  
  
    parameterBlock ref;  
    ref.x = 6;  
  
    byReference(&ref);  
  
    //Some Comment  
}
```