

Please complete the following questions. If you need help, feel free to ask questions both from other students and from me. If you cannot finish during the class time then you should finish this lab at home. After you finish, feel free to clean up (*i.e.*, remove) any files or directories that were created, you won't need them in the future. But be careful not to remove any other important files you may have in your home directory.

1. Preliminary.

- (a) Change your location to the 'cse399' directory you created in previous homeworks or labs (create it again if you deleted it). Copy the directory /home1/m/mcorliss/teaching/cse399/lab5 and all its subcontents to your current location. Change your location to this new directory (lab5).

2. Unix environment.

- (a) Set the PATH variable to '.:\$PATH'.

**Answer:**

```
bash$ export PATH=.:$PATH
```

- (b) Type 'ls' and hit enter. What happens? Use an absolute path to run the 'ls' that is in '/bin'.

**Answer:**

The 'ls' in the current directory is executed since the path '.' appears earlier in \$PATH than '/bin/'. If you run '/bin/ls' however, the 'ls' command for listing files and directories is executed, instead.

- (c) Set the PATH variable to its original value.

**Answer:**

```
bash$ export PATH=$(echo $PATH|sed -e 's/^[^:]*://')
```

- (d) Print out all environment variables.

**Answer:**

```
bash$ env
```

- (e) Print out just the environment variable LOGNAME.

**Answer:**

```
bash$ echo $LOGNAME
```

- (f) Add an environment variable FOO='foo' to your '.profile'.

**Answer:**

In '.profile', add a line: 'FOO=foo'.

- (g) Without logging out and logging back in, reload the '.profile' (after you finish the lab, you might want to remove the FOO line).

**Answer:**

```
bash$ source .profile
```

(h) On the command-line, alias 'pine' to 'goo'.

**Answer:**

```
bash$ alias pine='goo'
```

(i) Make this alias for 'pine' permanent (although after you finish this lab you will probably want to remove it).

**Answer:**

Add a line, 'alias pine=wrong' in '.bashrc'. Then source the '.profile' again.

### 3. Compiling C programs and Makefiles.

(a) Compile the program 'helloworld.c' using the second optimization level. You should compile in two steps: 1) compiling 'helloworld.c' to an object file, and 2) building an executable from that object file.

**Answer:**

```
bash$ gcc -O2 -c helloworld.c
```

```
bash$ gcc -O2 -o helloworld helloworld.o
```

(b) Build a simple Makefile for helloworld (without any variables or a clean rule).

(c) Add the following variables to your Makefile (described in the lectures): CC, LD, CFLAGS, LDFLAGS, OBJS, and PROG

(d) Add a clean rule for the Makefile.

(e) Finally, add wildcards to your makefile (although, it's not that useful in this case, since you only have one .c file). The rules in your Makefile should contain no filenames. Filenames should be stored only in variables.

**Answer:**

```
# Makefile:
```

```
# use "gcc" to compile source files.
```

```
CC = gcc
```

```
# the linker is also "gcc". Might not be for other compilers.
```

```
LD = gcc
```

```
# Compiler flags go here.
```

```
CFLAGS = -O2
```

```
# Linker flags go here. Currently there aren't any.
```

```
LDFLAGS =
```

```
# list of generated object files.
```

```
OBJS = helloworld.o
```

```
# program executable file name.
```

```
PROG = helloworld
```

```
# top-level rule to create the program.
```

```
all: $(PROG)
```

```
# linking the program
```

```
$(PROG): $(OBJS)
```

```
gcc $(LDFLAGS) -o $(PROG) $(OBJS)
```

```
# compiling the source file
```

```
%.o: %.c
```

```
gcc $(CFLAGS) -c $*.c
```

```
# cleaning everything that can be
```

```
# automatically created with "make"
```

```
clean:
```

```
bin/rm -f $(PROG) $(OBJS)
```