COP4600 Discussion 1

Introduction to Make

J David Smith
emailson@ufl.edu
What is This Session?
What is Minix?
What is Minix?

Minimal Unix
What is VMWare?
What is Make?
Structure of a Makefile

A Makefile is composed of **rules**.

When you run `make` the first rule is executed.

```makefile
CC=gcc

project: main.o foo.o
  $(CC) -o $@ $^

%.o: %.c
  $(CC) -c -o $@ $^

nothing:
  @echo "Nothing!"
  @echo "Really!"

.PHONY: nothing
```
Structure of a Rule

**project**: main.o foo.o

$(CC) -o $@ $^$

Each rule begins with the **target**. The target is the file that will be produced by the rule.
The target is followed by a space-separated list of dependencies.

Each dependency is either a file or a rule.

If a file is missing and a rule for it exists, it will be made.
Structure of a Rule

project: main.o foo.o
$(CC) -o $@ $^  

The dependencies are followed by one or more commands. Each command will be executed in turn. All commands must be indented consistently.
Structure of a Rule

project: main.o foo.o

$(CC) -o $@ $^$

Make also allows the use of variables.

Some commonly used variables are:

```
CC    C Compiler
CCFLAGS Flags (e.g. -Wall) for the C Compiler
```
Structure of a Rule

project: main.o foo.o
$(CC) -o $@ $^
Structure of a Rule

\%\texttt{.o:} \%\texttt{.c}

$(\texttt{CC})$ -c -o $\texttt{@}$ $\texttt{^}$

Make also supports wildcards (\%).

The string matching \% in the target is substituted for \% in the dependency list.

make \texttt{foo.o} would depend on \texttt{foo.c} in this example.
Structure of a Rule

nothing:
  @echo "Nothing!"
  @echo "Really!"

.PHONY: nothing

Not all rules produce files.

We add such rules to the special target .PHONY to inform make of this.
Exercise 1

Writing a Makefile
Setup

1. Log into a CISE machine via SSH
   Windows: PuTTY
   Linux & Mac: Terminal ssh

2. Clone the starter code
   
   ```bash
   git clone https://github.com/emallson/ex-make.git
   ```

3. Enter the directory
   
   ```bash
   cd ex-make
   ```
Tasks

1. Write a wildcard rule to compile .o files from .c

2. Write a rule `hello` that compiles the binary from the .o files

3. Write a rule `clean` that restores the directory to its initial state