

CS-344 - Unix Operating System Fundamentals

Lecture 13
GNU Make
&
Review

Based on slides created by
Dr. Bangalore for the
Spring 2005 offering of
the course

GNU Make (I)

- ❑ Automatic process of building (compile & link) programs
- ❑ Define rules that govern how the code is built
- ❑ Rules (or dependencies) are defined in a file generally called *Makefile*
- ❑ GNU Make process the *Makefile* and builds the project generating a final executable
- ❑ Make detect changes on source files and recompiles only what's necessary

GNU Make (II)

- Makefile rules define three things:
 1. The file that will be built when the rule is processed
 2. The process required to make files into the final product
 - To build a c project: compile the c files into an object file (.o) then link all the object files together
 3. The list of dependencies for each file. The dependencies must be created before the file can be processed. The dependencies can be arbitrary files or can refer to other rules

12/7/2005

4

Example

- | | |
|---|---|
| <ul style="list-style-type: none"> □ compute.c
 <code>extern int someglobal;
int computer(void) {
 return 5 * someglobal;
}</code> □ io.c
 <code>#include <stdio.h>
#include "myprogram.h"

int foo(void) {
 printf ("The value is: %d.\n",
 computer());
 return 1;
}</code> | <ul style="list-style-type: none"> □ init.c
 <code>#include <stdio.h>
#include "myprogram.h"

int someglobal = 11;

int main(void) {
 foo();
 return 0;
}</code> □ myprogram.h
 <code>int computer(void);
int foo(void);</code> |
|---|---|

12/7/2005

5

Simple Makefiles (I)

- | | |
|---|---|
| <p><u>1st Version</u></p> <pre># comments # "all" is the default target. # points to myprogram all: myprogram myprogram: io.o init.o compute.o gcc -o myprogram io.o init.o compute.o # Define the dependencies and compile # information for the three C files compute.o: compute.c gcc -Wall -c -o compute.o compute.c init.o: init.c myprogram.h gcc -Wall -c -o init.o init.c io.o: io.c myprogram.h gcc -Wall -c -o io.o io.c</pre> | <p><u>2nd Version</u></p> <pre>CC=gcc CFLAGS=-Wall COMPILE=\$(CC) \$(CFLAGS) -c all: myprogram myprogram: io.o init.o compute.o \$(CC) -o myprogram io.o init.o compute.o compute.o: compute.c \$(COMPILE) -o compute.o compute.c init.o: init.c myprogram.h \$(COMPILE) -o init.o init.c io.o: io.c myprogram.h \$(COMPILE) -o io.o io.c</pre> |
|---|---|

12/7/2005

6

Simple Makefiles (II)

3rd Version

```
CC=gcc
CFLAGS=-Wall
COMPILE=$(CC) $(CFLAGS) -c
```

```
all: myprogram
```

```
myprogram: io.o init.o compute.o
$(CC) -o myprogram io.o \ init.o
compute.o
```

```
%O: %.c
$(COMPILE) -o $@ $<
```

4th Version

```
CC=gcc
CFLAGS=-Wall
COMPILE=$(CC) $(CFLAGS) -c
```

```
all: myprogram
```

```
myprogram: io.o init.o compute.o
$(CC) -o myprogram io.o init.o \
compute.o
```

```
# Define a special dependency on a
# header file
init.o io.o: myprogram.h
```

```
%O: %.c
$(COMPILE) -o $@ $<
```

12/7/2005

7

Simple Makefiles (III)

Final Version

```
OBJS = io.o init.o compute.o
EXECUTABLE = myprogram
```

```
CC=gcc
CFLAGS=-Wall
COMPILE=$(CC) $(CFLAGS) -c
```

```
# "all" is the default target.
# Simply make it point to myprogram
```

```
all: $(EXECUTABLE)
```

```
# Define the components of the program,
# and how to link them together
# These components are defined as
# dependencies so they need to be
# up-to-date
```

```
$(EXECUTABLE): $(OBJS)
$(CC) -o $(EXECUTABLE) $(OBJS)
```

```
# Define a special dependency on a
# header file
```

```
init.o io.o: myprogram.h
```

```
# Specify that all .o files depend on .c
# files, and indicate how the c files
# are converted (compiled) to the .o files
```

```
%O: %.c
$(COMPILE) -o $@ $<
```

```
clean:
-rm $(OBJS) $(EXECUTABLE) *~
```

12/7/2005

8

Problem

- Calculate the total spent at the grocery store:

Code	Product	Quantity	Price
1234	Coke	2	1.10
9433	Windex	1	3.45
1534	Coffee	2	8.34
5943	Rice	5	1.33
2358	Butter	2	0.99

- Output:

Code	Product	Quantity	Price	ProductTotal
1234	Coke	2	1.10	2.20
9433	Windex	1	3.45	3.45
1534	Coffee	2	8.34	16.68
5943	Rice	5	1.33	6.65
2358	Butter	2	0.99	1.98
Total number of products bought: 12				
Subtotal:			30.96	
Tax:			3.71	
Total:			34.67	

12/7/2005

9

Solution 1 (II)

- ❑ The subtotal is the sum of all the product totals:
`subTotal=`echo "scale=2 $subTotal+$pTotal" | tr ' ' '\n' | bc``
- ❑ Add the column product Total to the output, in this case the output goes to the file "receipt"
`sed -n "/$product/s/$/\t$pTotal/ p" products >> receipt`
- ❑ Repeat this for each product in the grocery list

```
for product in $productList
do
...
done
```

12/7/2005

13

Solution 1 (III)

- ❑ Calculate the tax:
`tax=`echo "scale=2 $subTotal*$taxPercentage" | tr ' ' '\n' | bc``
- ❑ Calculate the total:
`total=`echo "scale=2 $subTotal+$tax" | tr ' ' '\n' | bc``
- ❑ Output the totals to *receipt*:

```
echo >> receipt
echo "Total number of products bought: $numProducts" >> receipt
echo "Subtotal:           $subTotal" >> receipt
echo "Tax:                $tax" >> receipt
echo "Total:              $total" >> receipt
```

12/7/2005

14

Solution 1 (IV)

- ❑ Some previous initialization steps:

```
productList=`sed -n '2~1 p' products | cut -f 1 | tr '\n' ' '`
if [ -f receipt ]; then rm receipt; fi
sed -n '/Code/s/$/\tProductTotal/ p' products > receipt
subTotal=0
tax=0
numProducts=0
total=0
taxPercentage=0.12
```

12/7/2005

15

Solution 2 (III)

- ❑ Finally, the receipt's total:


```
total=`echo "$subTotal+$tax " | tr ' ' '\n' | bc`
echo "Total:          $total" >> receipt2
```
- ❑ We also need to delete the temporary files:


```
rm temp1 temp2 temp3
```
- ❑ And some initializing steps at the beginning:


```
if [ -f receipt2 ]; then rm receipt2; fi

taxPercentage=0.12
```

12/7/2005

19

Solution 2

```
if [ -f receipt2 ]; then rm receipt2; fi

taxPercentage=0.12

# create a new column with the amount per product (including heading)
echo "Product Total" > temp1
sed -n 2-4 p products | cut -f3,4 | tr '\t' '\n' | bc > temp2
cat temp1 temp2 > temp3
paste products temp3 > receipt2

rm temp1 temp3 # delete temporal files

# add the final receipt info
echo >> receipt2
tempVar=`sed -n 2-4 p products | cut -f3 | tr '\n' '\t' | sed s/ /+//`
echo "Total number of products bought: $tempVar" | bc >> receipt2
subTotal=`cat temp2 | tr '\n' '\t' | sed -n s/ /+//n/ p | bc`
echo "Subtotal:          $subTotal" | bc >> receipt2
tax=`echo "$subTotal*$taxPercentage" | bc`
echo "Tax:              $tax" >> receipt2
total=`echo "$subTotal+$tax" | tr '\n' '\t' | bc`
echo "Total:           $total" >> receipt2

rm temp2 # delete temporal file
```

12/7/2005

20
