

# CS-333 - 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 detects changes on source files and recompiles only what's necessary

12/1/2006

3

### GNU Make (II)

- Makefile rules define three things:
  - The file that will be built when the rule is processed
  - 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
  - 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

12/1/2006

4

### Example

- compute.c**

```
extern int someglobal;
int computer(void) {
    return 5 * someglobal;
}
```
- io.c**

```
#include <stdio.h>
#include "myprogram.h"

int foo(void) {
    printf("The value is: %d.\n",
        computer());
    return 1;
}
```
- init.c**

```
#include <stdio.h>
#include "myprogram.h"

int someglobal = 11;

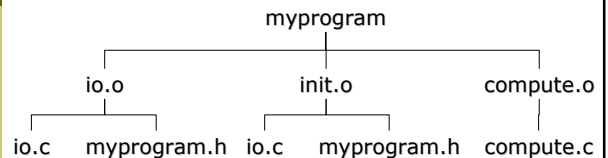
int main(void) {
    foo();
    return 0;
}
```
- myprogram.h**

```
int computer(void);
int foo(void);
```

12/1/2006

5

### Dependency tree



12/1/2006

6

## Simple Makefiles (I)

### 1st Version

```
# comments
# "all" is the default target. The make
# utility will execute this target if no
# other one is specified
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
```

12/1/2006

7

### 2nd 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

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
```

## 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 $@ $<
```

```
%.O: %.c states .O file can be built from a
corresponding .c
$@ is the name of the file to be made
$< the name of the related file that caused
the action
$? is the names of the changed dependents
```

12/1/2006

8

### 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 $@ $<
```

## 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
```

12/1/2006

9

```
$(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) *~
```

# Review...

## Through a problem

12/1/2006

10

## 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
<b>Total number of products bought: 12</b>				
<b>Subtotal:</b>			<b>30.96</b>	
<b>Tax:</b>			<b>3.71</b>	
<b>Total:</b>			<b>34.67</b>	

12/1/2006

11

## Solution

```
productList=$(sed -n 2-7 p products | cut -f 1 | tr '\n' ' ')
if [ -f receipt ]; then rm receipt fi

sed -n 7/Code/s/$(^ProductTotal/ 'p' products >> receipt

subTotal=0
tax=0
numProducts=0
numItems
taxPercentage=0.12

for product in $productList
do
expression=$(sed -n 7/Products/ 'p' products | cut -f 3,4 | tr '\n' ' ')
let numProducts=$numProducts+${#expression} | cut -d' ' -f 1
pTotal=$(echo "scale=2 $expression" | tr '\n' '\0' | bc)
subTotal=$(echo "scale=2 $subTotal+$pTotal" | tr '\n' '\0' | bc)
sed -n 7/Products/s/$(^SubTotal+$pTotal/ 'p' products >> receipt
done
tax=$(echo "scale=2 $subTotal*$taxPercentage" | tr '\n' '\0' | bc)
total=$(echo "scale=2 $subTotal+$tax" | tr '\n' '\0' | bc)
echo ">> receipt
echo "Total number of products bought: $numProducts" >> receipt
echo "Subtotal: $subTotal" >> receipt
echo "Tax: $tax" >> receipt
echo "Total: $total" >> receipt
```

12/1/2006

12

## Alternate Solution

```
if [ ! receipt ]; then rm receipt; fi
taxPercentage=0.12
# create a new column with the amount per product (including heading)
echo ProductTotal>temp1
sed -n '2~1' p' products | cut -f 3,4 | tr '\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 '2~1' p' products | cut -f 3 | tr '\n' '*' | sed 's/ /$/'
echo "Total number of products bought: (echo $tempVar | bc)" >> receipt2
subTotal= cat temp2 | tr '\n' '+' | sed -n 's/ /$/' | bc
echo "Subtotal: (echo $subTotal | bc)" >> receipt2
tax= echo "$subTotal*$taxPercentage" | tr '\n' '+' | bc
echo "Tax: (echo $tax)" >> receipt2
total= echo "$subTotal+$tax" | tr '\n' '+' | bc
echo "Total: (echo $total)" >> receipt2
rm temp2 # delete temporal file
```

## Solution 1 (I)

- Each line on the input file has the form:
 

Code	Product	Quantity	Price
1234	Coke	2	1.10
- The total per product is given by Quantity\*Price
- To Calculate the total per product:
 

```
expression=`sed -n '/$product/ p' products | cut -f 3,4 | tr '\t' '*'`
pTotal=`echo "scale=2 $expression" | tr '\n' '\n' | bc`
```
- We also need the amount of products bought:
 

```
let numProducts=$numProducts+' echo $expression | cut -d '*' -f 1'
```

## Solution 1 (II)

- The subtotal is the sum of all the product totals:
 

```
subTotal=`echo "scale=2 $subTotal+$pTotal" | tr '\n' '\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
```

## Solution 1 (III)

- Calculate the tax:
 

```
tax=`echo "scale=2 $subTotal*$taxPercentage" | tr '\n' '\n' | bc`
```
- Calculate the total:
 

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

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

## Solution 1 (IV)

- Some previous initialization steps:
 

```
productList=`sed -n '2~1' p' products | cut -f 1 | tr '\n' '\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
```

## Solution 1 (V)

```
productList=`sed -n '2~1' p' products | cut -f 1 | tr '\n' '\n'`
if [ ! receipt ]; then rm receipt; fi
sed -n '/Code/s/$/\tProductTotal/ p' products > receipt
subTotal=0
tax=0
numProducts=0
total=0
taxPercentage=0.12
for product in $productList
do
expression=`sed -n '/$product/ p' products | cut -f 3,4 | tr '\t' '*'`
let numProducts=$numProducts+' echo $expression | cut -d '*' -f 1'
pTotal=`echo "scale=2 $expression" | tr '\n' '\n' | bc`
subTotal=`echo "scale=2 $subTotal+$pTotal" | tr '\n' '\n' | bc`
sed -n '/$product/s/$/\t$pTotal/ p' products >> receipt
done
tax=`echo "scale=2 $subTotal*$taxPercentage" | tr '\n' '\n' | bc`
total=`echo "scale=2 $subTotal+$tax" | tr '\n' '\n' | bc`
echo >> receipt
echo "Total number of products bought: $numProducts" >> receipt
echo "Subtotal: (echo $subTotal)" >> receipt
echo "Tax: (echo $tax)" >> receipt
echo "Total: (echo $total)" >> receipt
```

## Solution 2 (I)

- Each line on the input file has the form:
 

Code	Product	Quantity	Price
1234	Coke	2	1.10
- create a new column with the total per product (including heading) :
 

```
echo ProductTotal > temp1
sed -n '2~1 p' products | cut -f 3,4 | tr '\t' '*' | bc > temp2
cat temp1 temp2 > temp3
paste products temp3 > receipt2
```

12/1/2006

19

## Solution 2 (II)

- Add the final receipt info:
 

```
echo >> receipt2
tempVar=`sed -n '2~1 p' products | cut -f 3 | tr '\n' '+' | sed 's/+$//'`
echo "Total number of products bought: `echo $tempVar | bc`" >> receipt2
```
- Subtotal:
 

```
subTotal=`cat temp2 | tr '\n' '+' | sed -n 's/+$/\n/ p' | bc`
echo "Subtotal: `echo $subTotal | bc`" >> receipt2
```
- Tax:
 

```
tax=`echo "$subTotal*$taxPercentage" | tr '\n' '+' | bc`
echo "Tax: `echo $tax`" >> receipt2
```

12/1/2006

20

## Solution 2 (III)

- Finally, the receipt's total:
 

```
total=`echo "$subTotal+$tax" | tr '\n' '+' | bc`
echo "Total: `echo $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
```

12/1/2006

21

## Solution 2

```
if [ -f receipt2 ]; then rm receipt2; fi
taxPercentage=0.12
# create a new column with the amount per product (including heading)
echo ProductTotal > temp1
sed -n '2~1 p' products | cut -f 3,4 | tr '\t' '*' | bc > temp2
cat temp1 temp2 > temp3
paste products temp3 > receipt2
rm temp1 temp2 # delete temporary files
# add the final receipt info
echo >> receipt2
tempVar=`sed -n '2~1 p' products | cut -f 3 | tr '\n' '+' | sed 's/+$//'`
echo "Total number of products bought: `echo $tempVar | bc`" >> receipt2
subTotal=`cat temp2 | tr '\n' '+' | sed -n 's/+$/\n/ p' | bc`
echo "Subtotal: `echo $subTotal | bc`" >> receipt2
tax=`echo "$subTotal*$taxPercentage" | tr '\n' '+' | bc`
echo "Tax: `echo $tax`" >> receipt2
total=`echo "$subTotal+$tax" | tr '\n' '+' | bc`
echo "Total: `echo $total`" >> receipt2
rm temp2 # delete temporary file
```

12/1/2006

22