CS213: Software Systems Laboratory

Autumn 2023-24

Koteswararao Kondepu

k.kondepu@iitdh.ac.in

Recap *

$# $@ $* $?

- Special Variables ✓ $1 $2 . . .
- Process management ✓
- Directory and File operations ✓
- Loops

ps -aux | grep r ?

PID

pkill -9 PID

kill -9 ✓

touch
mkdir
rm -r
cd
chmod -r ✓

if ... else ... fi
if .. else elif .. fi
for ... do stmt .. done,
while .. do ... done

[[ ]]

$(( )),

( ) ✓

Outline

- ✓ Bash commands: sed, awk, tar
- Makefile, libraries and linking ✓
- * Networking commands:
  - ping, traceroute, ifconfig, netstat, curl, wget, tcpdump, ssh, scp, rsync
- Startup scripts

Sed - Stream editor

Performs functions on file like searching, find and replace, insertion or deletion.

Usage: **sed <options> <script> <file>**

Common Options:

-i : To edit the source file directly

-e: To combine multiple sed commands using a single call

-f : finds all rules that are applied in a specific script file

g: To replace all the occurrences of a string in a file globally

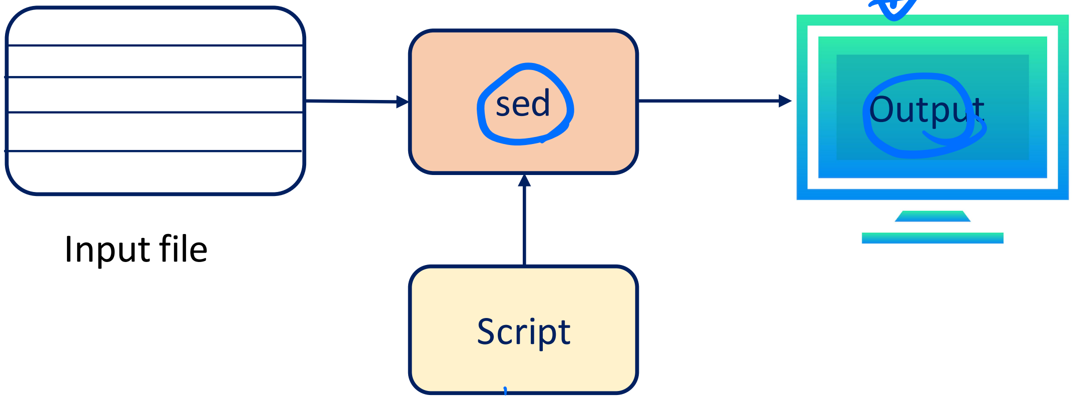
^: Special character to match the beginning of the regular expression

n : Suppresses the output

s : Substitution operation

y : Transforming characters

d : delete



```
#!/bin/bash
sed ‘s/pettern/pettern1’cs213.txt
sed ‘y/abcd/efgh’ cs213.txt
```

Sed: Examples

linux.txt

```
unix is great os. unix is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

sed_example1.sh

```
#!/bin/bash
sed 's/unix/linux/' linux.txt
```

output

```
linux is great os. unix is opensource. unix is free os.
learn operating system.
linux linux which one you choose.
linux is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful
```

Here the **"s"** specifies the substitution operation. The **"/"** are delimiters. The **"unix" is the search pattern** and the **"linux" is the replacement string.** By default, the sed command replaces the first occurrence of the pattern in each line.

sed_example2.sh

```
#!/bin/bash
sed 's/unix/linux/g' linux.txt
```

output

```
linux is great os. linux is opensource. linux is free os.
learn operating system.
linux linux which one you choose.
linux is easy to learn.linux is a multiuser os.Learn linux .linux is a powerful.
```

Replaces all occurrence of the word "unix" with "linux"

Sed: Examples (1)

linux.txt

```
unix is great os. unix is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

sed_example3.sh

```bash
#!/bin/bash
sed 's/unix/linux/2' linux.txt
```

output

```
unix is great os. linux is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.linux is a multiuser os.Learn unix .unix is a powerful.
```

- Replaces the second occurrence of the word “unix” with “linux” in a line

sed_example4.sh

```bash
#!/bin/bash
echo "Welcome To The CS213 SSL" | sed 's/\(\b[A-Z]\)/\(\1\)/g'
```

output

```
(W)elcome (T)o (T)he (C)S213 (S)SL
```

- Apply the first character of every word in parenthesis

awk [Aho, Weinberger, and Kernighan]

A scripting language used for manipulating data and generating reports.

Usage: **awk 'selection _criteria {action }' input-file > output-file**

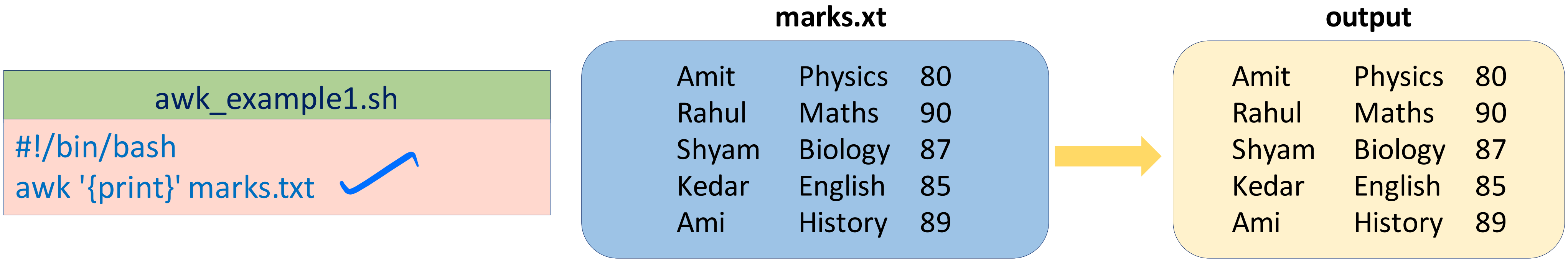
Common Options:

-f program files : Reads the source code of the script written on the awk command

-F fs: used as the input field separator

```
#!/bin/bash
awk ‘{print}’ cs213.txt
```

AWK: Examples



awk_example1.sh

```
#!/bin/bash
awk '{print}' marks.txt
```

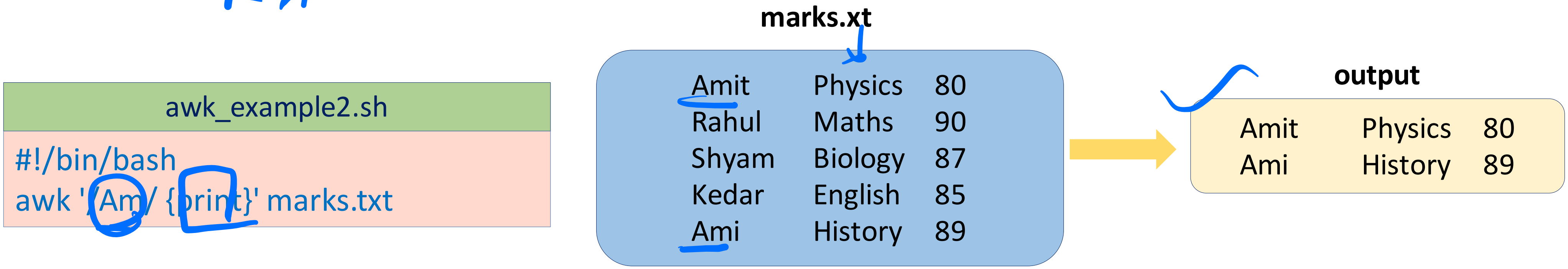
marks.xt

```
Amit     Physics   80
Rahul    Maths     90
Shyam    Biology   87
Kedar    English   85
Ami      History   89
```

output

```
Amit     Physics   80
Rahul    Maths     90
Shyam    Biology   87
Kedar    English   85
Ami      History   89
```

- No pattern is given. Thus, the actions are applicable to all the lines.
- Action *print* without any argument prints the whole line by default, so it prints all the lines of the file without failure.



awk_example2.sh

```
#!/bin/bash
awk '/Am/ {print}' marks.txt
```

marks.xt

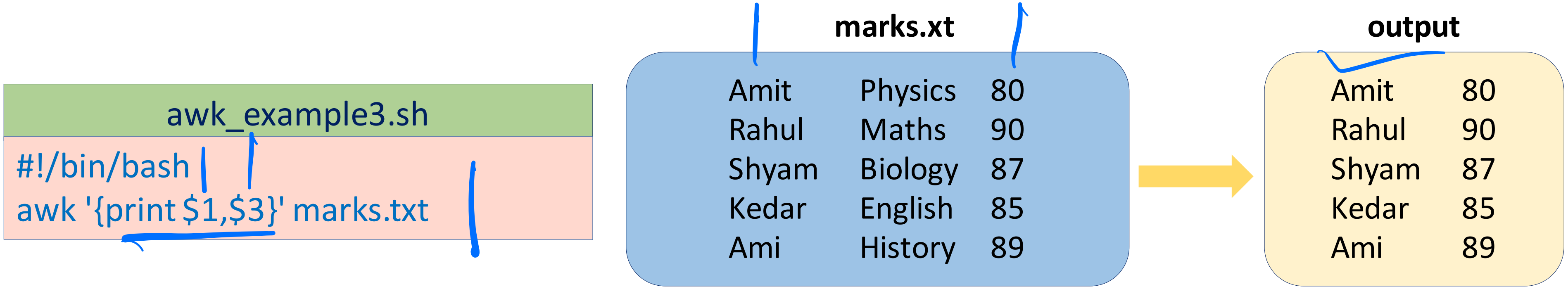
```
Amit     Physics   80
Rahul    Maths     90
Shyam    Biology   87
Kedar    English   85
Ami      History   89
```

output

```
Amit     Physics   80
Ami      History   89
```

- The awk command prints all the line which matches with the ‘Am’.

AWK: Examples (1)



awk_example3.sh

```
#!/bin/bash
awk '{print $1,$3}' marks.txt
```

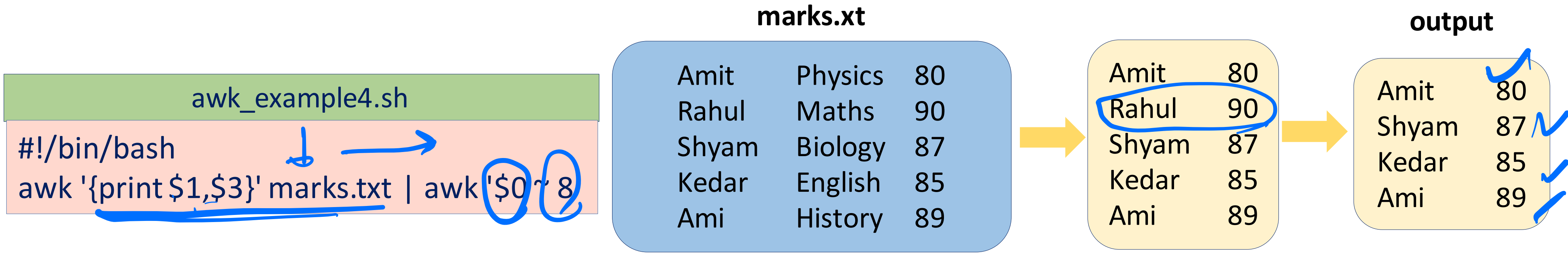
marks.xt

```
Amit     Physics   80
Rahul    Maths     90
Shyam    Biology   87
Kedar    English   85
Ami      History   89
```

output

```
Amit     80
Rahul    90
Shyam    87
Kedar    85
Ami      89
```

- The awk command splits the record delimited by whitespace character by default and stores it in the $n variables.
- If the line has 3 words, it will be stored in $1, $2 and $3 respectively. Also, $0 represents the whole line.



awk_example4.sh

```
#!/bin/bash
awk '{print $1,$3}' marks.txt | awk '$0 ~ 8
```

marks.xt

```
Amit     Physics   80
Rahul    Maths     90
Shyam    Biology   87
Kedar    English   85
Ami      History   89
```

```
Amit     80
Rahul    90
Shyam    87
Kedar    85
Ami      89
```

output

```
Amit     80
Shyam    87
Kedar    85
Ami      89
```

- ~. It looks for a field that contains the match string. The command example prints the lines that contain the pattern 8.

tar

- To create and manipulate streaming archive of files

Usage: **tar <options> <file> <patterns>**

-gz2

Common Options:

-c : To create an archive file

-x : Extract to disk from archive

-t : List contents of archive

-z : Filter the archive through gzip

-j : Filter the archive through bzip2

-v : Verbosely list files processed

-f : Read the archive from or write the archive to the specified file

tar -xzvf

tar_check.sh

```
#!/bin/bash
tar -cvzf cs213.tar.gz ${HOME}/*
```

zip

uzip

- To compress files to reduce file size and is also used as a file package utility. Zip is available in many operating systems like Unix, Linux, windows, etc.

Usage: **zip [options] [file_name.zip] [files_names]**

Common Options:

-d : To remove specific files from a zip archive

-u : Update files in the archive

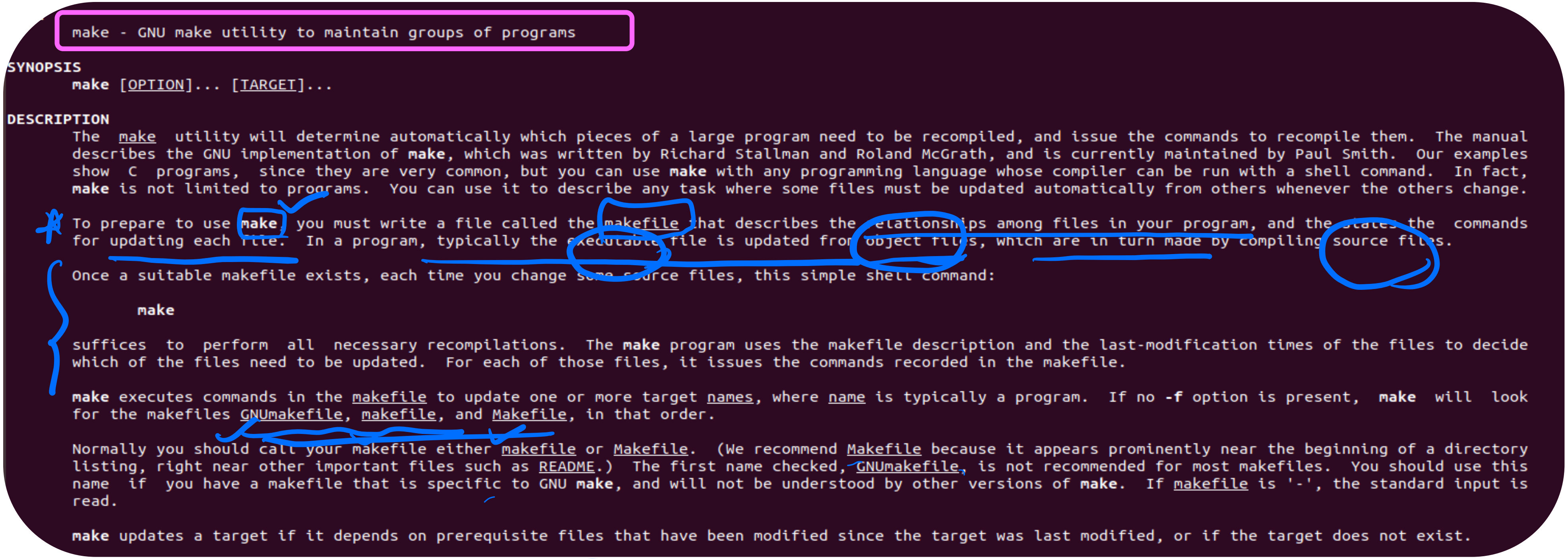
-r : Recursively zip a directory

tar_check.sh

```
#!/bin/bash
zip -r cs213.zip
```

Make utility

Command line utility used to process the instructions written in *Makefile*.



```
       make - GNU make utility to maintain groups of programs

SYNOPSIS
       make [OPTION]... [TARGET]...

DESCRIPTION
       The  make  utility will determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them.  The manual
       describes the GNU implementation of make, which was written by Richard Stallman and Roland McGrath, and is currently maintained by Paul Smith.  Our examples
       show  C  programs,  since they are very common, but you can use make with any programming language whose compiler can be run with a shell command.  In fact,
       make is not limited to programs.  You can use it to describe any task where some files must be updated automatically from others whenever the others change.

       To prepare to use make, you must write a file called the makefile that describes the relationships among files in your program, and the states the  commands
       for updating each file.  In a program, typically the executable file is updated from object files, which are in turn made by compiling source files.

       Once a suitable makefile exists, each time you change some source files, this simple shell command:

              make

       suffices  to  perform  all  necessary recompilations.  The make program uses the makefile description and the last-modification times of the files to decide
       which of the files need to be updated.  For each of those files, it issues the commands recorded in the makefile.

       make executes commands in the makefile to update one or more target names, where name is typically a program.  If no -f option is present,  make  will  look
       for the makefiles GNUmakefile, makefile, and Makefile, in that order.

       Normally you should call your makefile either makefile or Makefile.  (We recommend Makefile because it appears prominently near the beginning of a directory
       listing, right near other important files such as README.)  The first name checked, GNUmakefile, is not recommended for most makefiles.  You should use this
       name  if  you have a makefile that is specific to GNU make, and will not be understood by other versions of make.  If makefile is '-', the standard input is
       read.

       make updates a target if it depends on prerequisite files that have been modified since the target was last modified, or if the target does not exist.
```

Usage: make **<options> <target>**   Makefile ✓

Common Options:

- **-f**: used to use a file as a makefile
- **-d**: print the debugging information along with normal processing
- **-i**: used to ignore all errors in commands

```
#!/bin/bash
make
```

$ make   Makefile1

Makefile

make -f Makefile1

- *Makefile* is a script that automates compiling and linking.
- Essential in software development for efficient build.
- Commonly used to build C/C++ targets.

1 target: dependencies → $ gcc hello.c -o hello

- target: What to build; e.g., an executable
- dependencies: Files needed for building

hel : hello.c.

2 <Tab> command (or recipe)

- command: Action to perform the build and generate the executable
  - <compiler> <flags> <source files> -o <target>

gcc hello.c -O hello

Dont forget to include a <tab> character before every target recipe!

Why Makefile?

- **For example, 4 source files required to compile at the same time; main.c, hello.c, action.c and functions.h**

Example: gcc main.c hello.c action.c -o executable

Note

But if there are 1000s of source code files to be compiled at the same time, do we need to write manually all the files one by one ?

NO!

Simple Solution: With make and Makefile you only need to execute **make**

Common Makefile Macros

Variable	***Variable Description***
$@	The name of the executable file to be made
$?	The name of the changed dependencies
CC	Program to compile C programs, default is `gcc’
CXX	Program to compile C programs, default is `g++’
CFLAGS	Extra flags to be passed to C compiler
CXXFLAGS	Extra flags to be passed to C++ compiler
LDFLAGS	Extra flags to give to compilers when they are supposed to invoke the linker

Create executbale using Makefile Macros

```
CC = gcc
CFLAGS = -Wall -O2
myprogram: main.c a.c
$(CC) $(CFLAGS) -o $@ main.c  a.c
```

- **<target: dependencies>**
- **<compiler> <flags> -o <target> <source files>**

PS:

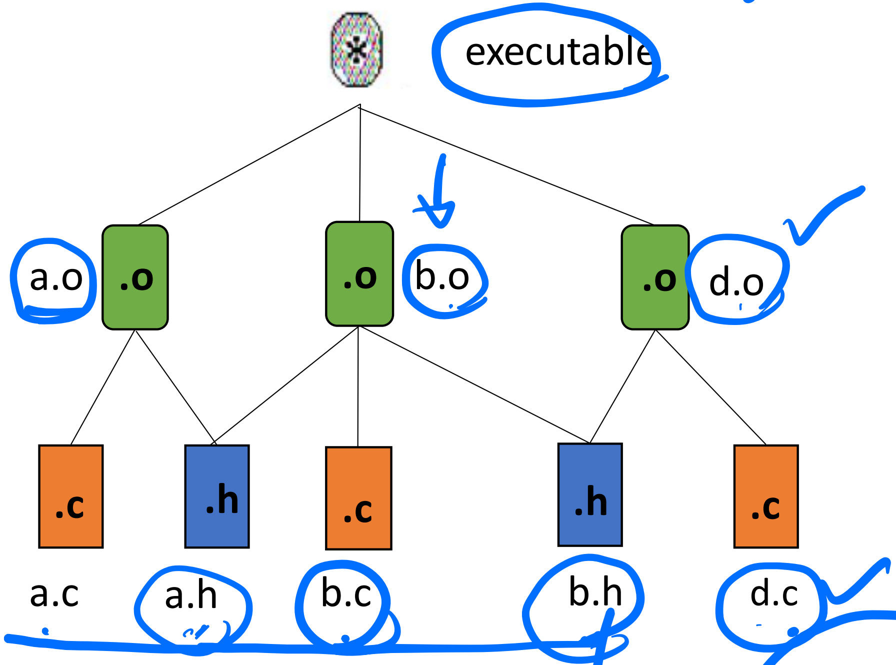
-wall : Stands for “Warning all” and instructs the compiler to enable a comprehensive set of warning messages.

-O2 : Stands for “Optimization Level 2” and instructs the compiler to apply a set of optimizations to the generated machine code

Writing a Makefile with dependencies

< .h>

" ←→ .h"



executable file , objective file source file

Sample Makefile

```
executable: a.o b.o d.o
        gcc a.o b.o  d.o -o executable
a.o: a.c a.h
        gcc -c a.c
b.o: a.h b.h b.c
        gcc  -c b.c
d.o: b.h d.c
        gcc  -c d.c
```

-o a.o

Rules and Targets in Makefile

- Rules define how to build targets
- Rules include prerequisites and a recipe
- Prerequisites are the dependencies for the target
- Recipe contains the commands to build the target

Example:

```
all: target
target: main.c helper.c
gcc -o target main.c helper.c
```

```
<rules>:  target/custom targets
target: <prerequisites>
gcc -o target main.c helper.c     ← <recipe>
```

Sample Makefile

make clean

CC = gcc → specifies the C compiler

CFLAGS = -Wall -O2 → holds the compilation flags, including `-Wall’ for warnings and `-O2’ for optimizations

LDLIBS = -lm → specifies any libraries that needs to be linked, here `-lm’ which is the math library

SOURCES = main.c helper.c → lists the source files

TARGET = myprogram → hold the name of the executable

all: $(TARGET) → this target depends on $(TARGET) which will be built if necessary and builds program from scratch if run make all

$(TARGET): $(SOURCES) → rpecifies the compilation rule to build the target
$(CC) $(CFLAGS) -o $(TARGET) $(SOURCES) $(LDLIBS)

clean: → removes the generated files if run make clean
rm -f $(TARGET)

rm -f *.o

.PHONY: all clean → lists targets that don’t represent files to be built like ‘all’ and ‘clean’

Makefile: Example#1

[Makefile – [illegible]]

Filename: ***Makefile***

```
# create [redacted] folder in Documents folder where all your source files contain
CC = gcc
CFLAGS = -I/home/user/Document/Makefile-Examples/Makefile1
hello: client.c server.c hello.h
	$(CC) $(CFLAGS) -o $@ client.c server.c
```

-I ✓

Execution: $ make –f ***Makefile***

Makefile: Example#2

Filename: *Makefile*

```
# create ~~CSL~~ folder in Documents folder where all your source files contain
CC = gcc
CFLAGS = -I/home/user/Document/Makefile-Examples/Makefile2
hello: main.c factorial.c hello.c functions.h
        $(CC) $(CFLAGS) -o $@  main.c factorial.c hello.c
```

Execution: $ make –f *Makefile*

Observations: (i) observe the output; (ii) try to remake and observe the console message;

'Executable' is upto date → this can be observed when there is no change in the source files

Makefile: Example#3

Filename: *Makefile*

```
CC = gcc
CFLAGS = -I/home/user/Document/Makefile-Examples/Makefile3

TARGET = result
all: $(TARGET)
$(TARGET): main.c functions.c hello.c functions.h
        $(CC) $(CFLAGS) -o $@ main.c functions.c hello.c ~~functions.h~~
clean:
        rm -f  $(TARGET)

.PHONY: all clean
```

Execution: $ make –f *Makefile*

```
$ make or $ make all
$ make clean
```

Makefile: Example#4

Filename: *Makefile*

```
CC = gcc
CFLAGS += -Wall   -I
CFLAGS += -I/home/user/Document/Makefile-Examples/Makefile4
TARGET = math_program
LDLIBS = -lm
all: $(TARGET)
$(TARGET): main.o hello.o
        $(CC) main.o hello.o -o $@ $(LDLIBS)
main.o: main.c functions.h
        $(CC) -c  $(CFLAGS)  main.c $(LDLIBS)
hello.o: hello.c  - Wall  -I
        $(CC) -c  $(CFLAGS) hello.c
clean:            - wall  -I
        rm -f  *.o $(TARGET)
.PHONY: all clean
```

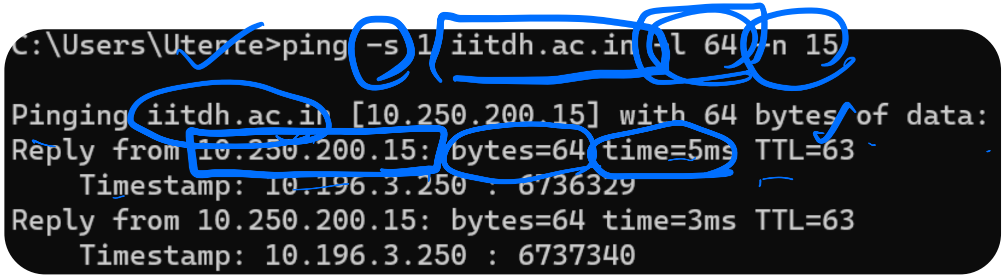
Observations: (i) comment LDLIBS, observe the console message; (ii) remove *.o in the clean target, and try to change the code in source file and remake, observe the different, if any

Execution: $ make –f *Makefile*

Networking Commands: ping [packet internet groper]

- Tests the connection between the local machine and the host server.
- Takes as input the IP address or the URL and sends a ICMP ECHO messages to the specified address with the message “PING” and get a response from the remote host or gateway
- Sends a small amount of data to the host server, and the host server replies to the computer

PING [-s] {IP_address | host_name} [size] [quantity]



```
C:\Users\Utente>ping -s 1 iitdh.ac.in -l 64 -n 15

Pinging iitdh.ac.in [10.250.200.15] with 64 bytes of data:
Reply from 10.250.200.15: bytes=64 time=5ms TTL=63
    Timestamp: 10.196.3.250 : 6736329
Reply from 10.250.200.15: bytes=64 time=3ms TTL=63
    Timestamp: 10.196.3.250 : 6737340
```

Usage: **ping <options> <destination>**

Common Options:

-a : Resolves the hostname to the respective IP address

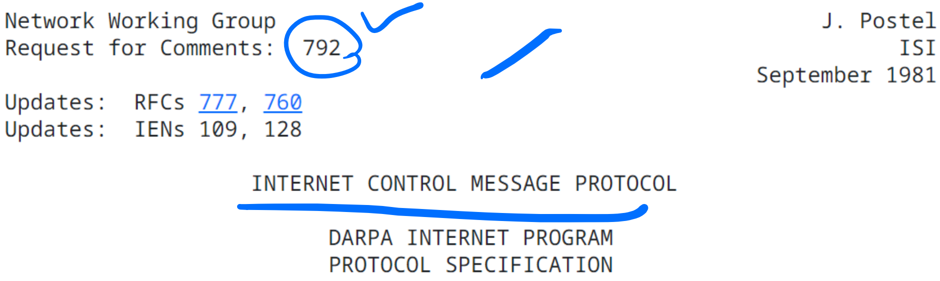
-w : Sets the timeout, the time after which the data packet will be rejected for each ping

-I : Interface from where the PING should be initiated

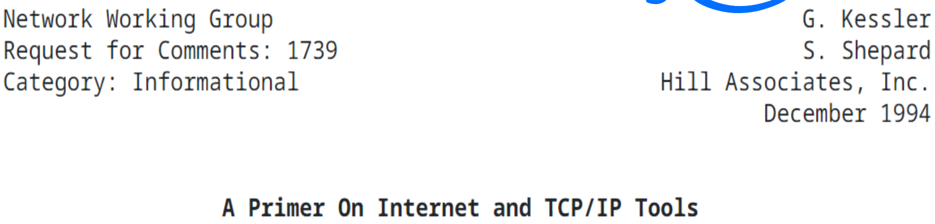
-n: Number of ICMP request

-U : Print user-to-user latency

Ping -I IP/



```
Network Working Group                                          J. Postel
Request for Comments:  792                                           ISI
                                                          September 1981
Updates:  RFCs 777, 760
Updates:  IENs 109, 128

                   INTERNET CONTROL MESSAGE PROTOCOL

                         DARPA INTERNET PROGRAM
                         PROTOCOL SPECIFICATION
```



```
Network Working Group                                     G. Kessler
Request for Comments: 1739                                S. Shepard
Category: Informational                        Hill Associates, Inc.
                                                       December 1994


              A Primer On Internet and TCP/IP Tools
```

Networking Commands: ping [packet internet groper]

- Tests the connection between the local machine and the host server.
- Takes as input the IP address or the URL and sends a ICMP ECHO messages to the specified address with the message “PING” and get a response from the remote host or gateway
- Sends a small amount of data to the host server, and the host server replies to the computer

PING [-s] {IP_address | host_name} [size] [quantity]

Usage: **ping <options> <destination>**

Common Options:

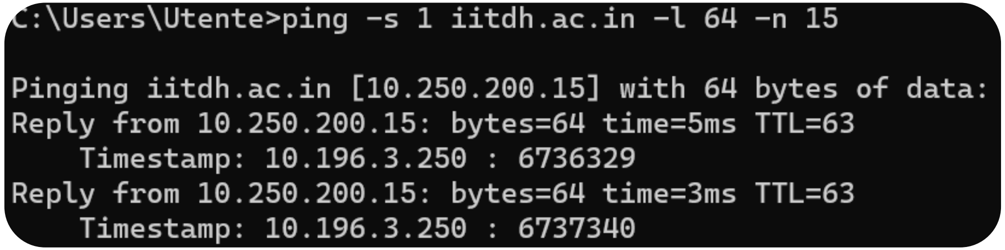
-a : Resolves the hostname to the respective IP address

-w : Sets the timeout, the time after which the data packet will be rejected for each ping

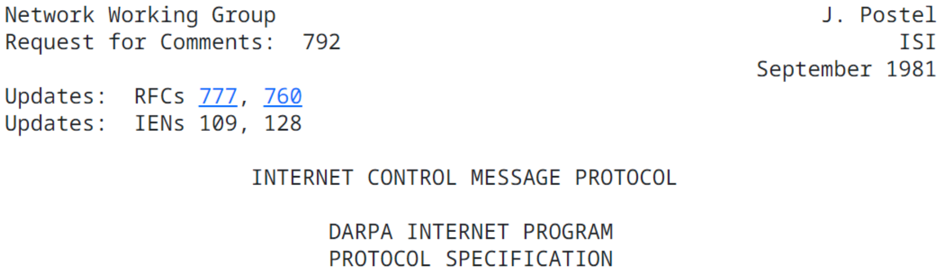
-I : Interface from where the PING should be initiated

-n: Number of ICMP request

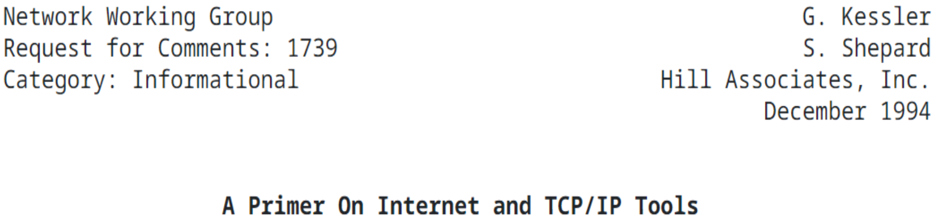
-U : Print user-to-user latency



```
C:\Users\Utente>ping -s 1 iitdh.ac.in -l 64 -n 15

Pinging iitdh.ac.in [10.250.200.15] with 64 bytes of data:
Reply from 10.250.200.15: bytes=64 time=5ms TTL=63
    Timestamp: 10.196.3.250 : 6736329
Reply from 10.250.200.15: bytes=64 time=3ms TTL=63
    Timestamp: 10.196.3.250 : 6737340
```

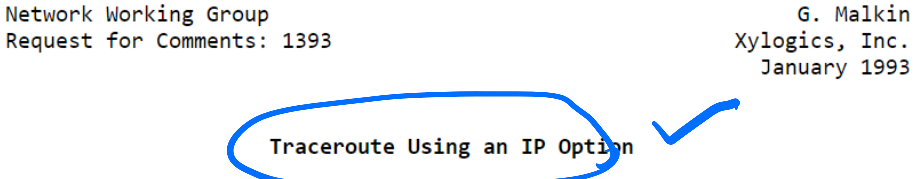


```
Network Working Group                                          J. Postel
Request for Comments:  792                                           ISI
                                                          September 1981
Updates:  RFCs 777, 760
Updates:  IENs 109, 128

                    INTERNET CONTROL MESSAGE PROTOCOL

                        DARPA INTERNET PROGRAM
                        PROTOCOL SPECIFICATION
```



```
Network Working Group                                         G. Kessler
Request for Comments: 1739                                    S. Shepard
Category: Informational                            Hill Associates, Inc.
                                                           December 1994


               A Primer On Internet and TCP/IP Tools
```

Networking Commands: traceroute



```
Network Working Group                                          G. Malkin
Request for Comments: 1393                                Xylogics, Inc.
                                                            January 1993


                  Traceroute Using an IP Option
```

- Traces the route from a computer to a host server
- Traces the connection for a fixed maximum number of hops
- Useful when you want to know about the route and about all the hops that a packet takes and can help in network
- *Troubleshooting*

Usage: **traceroute <options> <destination>**

Common Options:

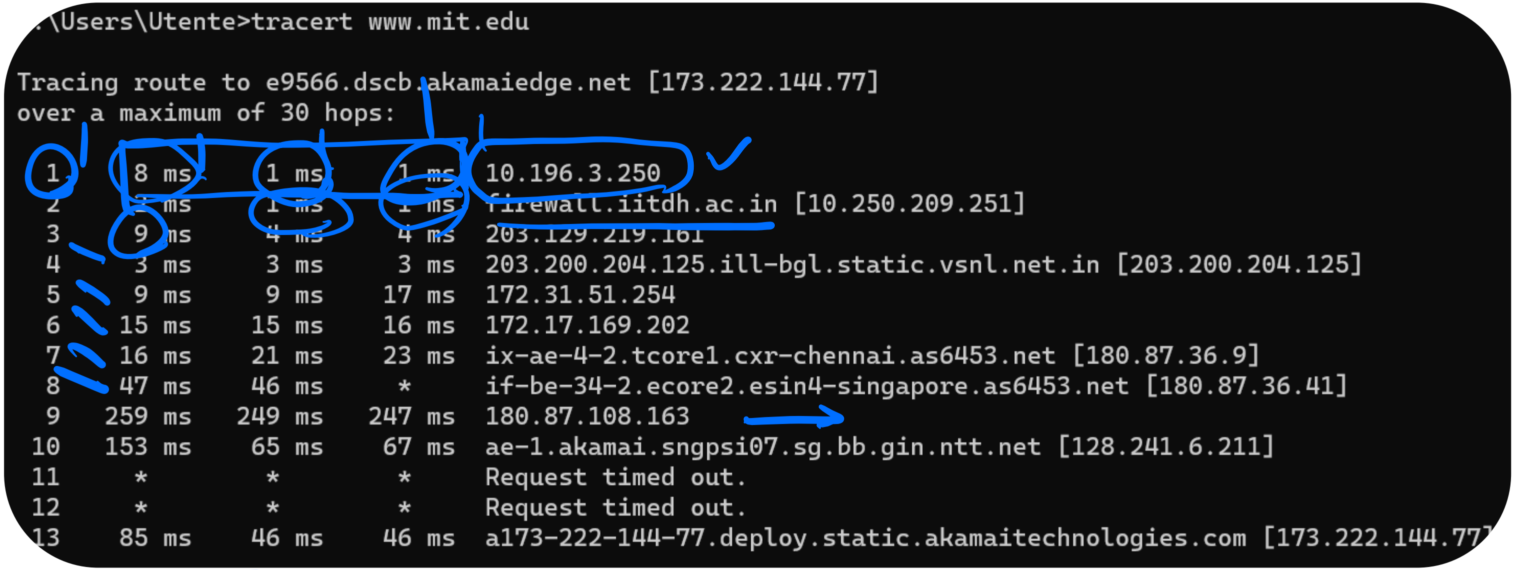
-d : Tells the tracert not no resolve the IP addresses to hostnames.

-h : Sets the maximum number of hops for which the tracert command will trace the connection.

-w : Sets the timeout time for each reply.

traceroute_check.sh

```
#!/bin/bash
traceroute -h 20 www.google.com
```



```
:\Users\Utente>tracert www.mit.edu

Tracing route to e9566.dscb.akamaiedge.net [173.222.144.77]
over a maximum of 30 hops:

  1     8 ms     1 ms     1 ms  10.196.3.250
  2     2 ms     1 ms     1 ms  firewall.iitdh.ac.in [10.250.209.251]
  3     9 ms     4 ms     4 ms  203.129.219.161
  4     3 ms     3 ms     3 ms  203.200.204.125.ill-bgl.static.vsnl.net.in [203.200.204.125]
  5     9 ms     9 ms    17 ms  172.31.51.254
  6    15 ms    15 ms    16 ms  172.17.169.202
  7    16 ms    21 ms    23 ms  ix-ae-4-2.tcore1.cxr-chennai.as6453.net [180.87.36.9]
  8    47 ms    46 ms     *     if-be-34-2.ecore2.esin4-singapore.as6453.net [180.87.36.41]
  9   259 ms   249 ms   247 ms  180.87.108.163
 10   153 ms    65 ms    67 ms  ae-1.akamai.sngpsi07.sg.bb.gin.ntt.net [128.241.6.211]
 11     *        *        *     Request timed out.
 12     *        *        *     Request timed out.
 13    85 ms    46 ms    46 ms  a173-222-144-77.deploy.static.akamaitechnologies.com [173.222.144.77
```

Networking Commands: ifconfig [interface configurator]

- Used to configure the kernel-resident network interfaces
- Used at the boot time to set up the interfaces as necessary
- Used to display the route and network interfaces

ifconf

Usage: **ifconfig [...OPTIONS] [INTERFACE]**

Common Options:

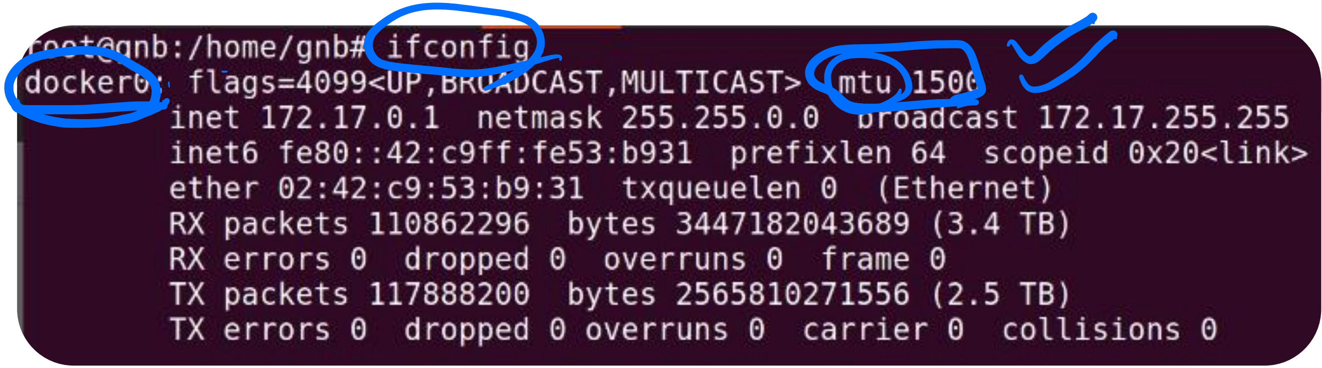
-a : Display all interfaces which are currently available, even if down.

-s : Display a short list.

mtu N : Sets the Maximum Transfer Unit (MTU) of an interface.

ifconfig [-v] [-a] [-s] [interface]

ifconfig [-v] interface [aftype] options | address ...



```
root@gnb:/home/gnb# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:c9ff:fe53:b931  prefixlen 64  scopeid 0x20<link>
        ether 02:42:c9:53:b9:31  txqueuelen 0  (Ethernet)
        RX packets 110862296  bytes 3447182043689 (3.4 TB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 117888200  bytes 2565810271556 (2.5 TB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ifconfig_check.sh

```
#!/bin/bash
ifconfig -a
```

ifcon 1 → mut < 9600 >
400

Networking Commands: netstat [network statistics]

- Displays various network related information such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc

Usage: **netstat [...OPTIONS]**

Common Options:

-p : Displays the programs associated with the open socket.

-s : Gives detailed statistics of all the ports.

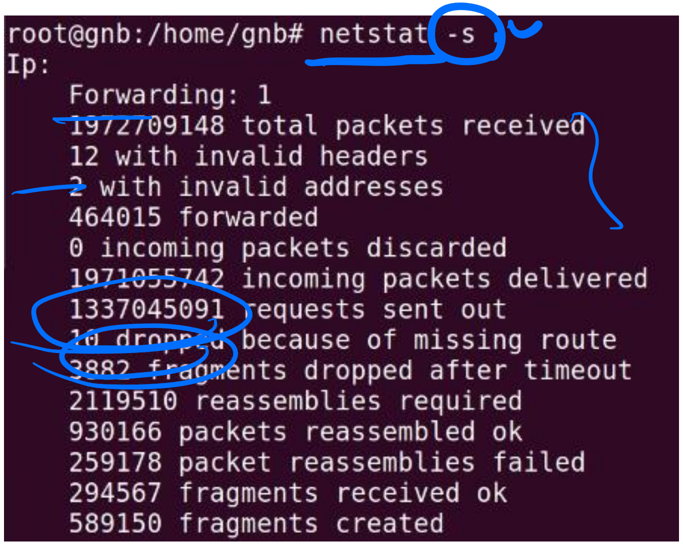
-r : Gives information related to the routing table.

-a : shows the state of all sockets.

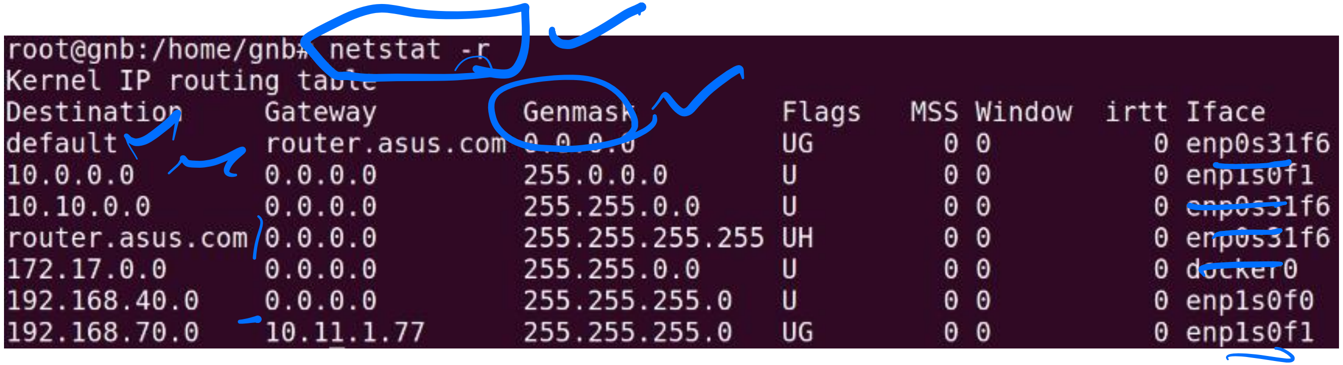
-l : List only the listening ports

netstat_check.sh
#!/bin/bash netstat -a

netstat -r



```
root@gnb:/home/gnb# netstat -s
Ip:
    Forwarding: 1
    1972709148 total packets received
    12 with invalid headers
    2 with invalid addresses
    464015 forwarded
    0 incoming packets discarded
    1971055742 incoming packets delivered
    1337045091 requests sent out
    10 dropped because of missing route
    3882 fragments dropped after timeout
    2119510 reassemblies required
    930166 packets reassembled ok
    259178 packet reassemblies failed
    294567 fragments received ok
    589150 fragments created
```



```
root@gnb:/home/gnb# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         router.asus.com 0.0.0.0         UG        0 0          0 enp0s31f6
10.0.0.0        0.0.0.0         255.0.0.0       U         0 0          0 enp1s0f1
10.10.0.0       0.0.0.0         255.255.0.0     U         0 0          0 enp0s31f6
router.asus.com 0.0.0.0         255.255.255.255 UH        0 0          0 enp0s31f6
172.17.0.0      0.0.0.0         255.255.0.0     U         0 0          0 docker0
192.168.40.0    0.0.0.0         255.255.255.0   U         0 0          0 enp1s0f0
192.168.70.0    10.11.1.77      255.255.255.0   UG        0 0          0 enp1s0f1
```

Networking Commands: curl and wget

- Used in downloading files from the internet through CLI
- Preferred for automation since it is designed to work without user interaction
- It can transfer multiple files at once.
- Poweredby Libcurl

Usage: **curl -O <fileLink>** (or) **wget <fileLink>**

```
curl -O https://github.com/curl/curl/blob/master/README.md
wget https://github.com/curl/curl/archive/refs/heads/master.zip
```

Command Options:

-o : saves the downloaded file on the local machine with the name provided in the parameters

-O : downloads the file and saves it with the same name as in the URL

-C : Resumes download which has been stopped due to some reason.

Networking Commands: tcpdump ✓

- Captures the traffic that is passing through the network interface and displays it.
- Can be used as a security tool as well.
- It saves the captured information in a pcap file which can then be opened through Wireshark or through the command tool itself

tcpdump_check.sh

```
#!/bin/bash
tcpdump -w 0001.pcap -i eth0
```

Usage: **tcpdump -i <network_device>**

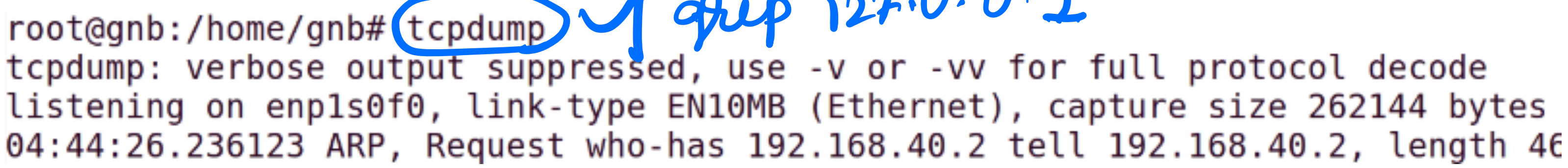
Common Options:

-i : Interface

-c : Captures a specified number of packets.

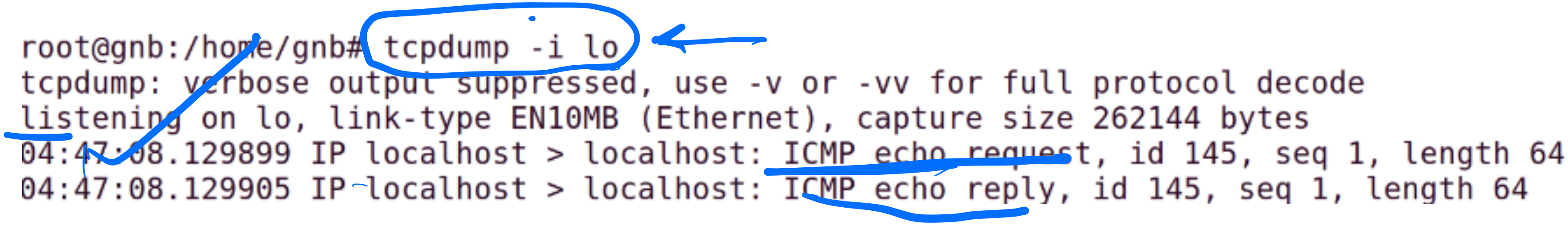
-w: To capture and save the file in a .pcap format

-r : To read captured packets from a file



```
root@gnb:/home/gnb# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp1s0f0, link-type EN10MB (Ethernet), capture size 262144 bytes
04:44:26.236123 ARP, Request who-has 192.168.40.2 tell 192.168.40.2, length 46
```

| grep 127.0.0.1

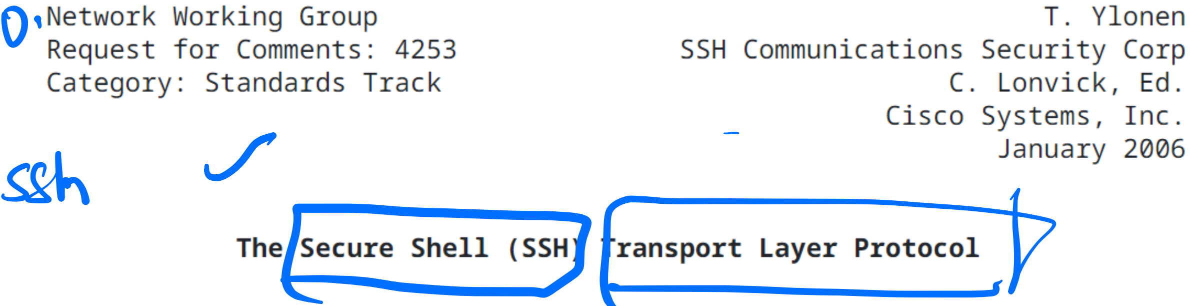


```
root@gnb:/home/gnb# tcpdump -i lo
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
04:47:08.129899 IP localhost > localhost: ICMP echo request, id 145, seq 1, length 64
04:47:08.129905 IP localhost > localhost: ICMP echo reply, id 145, seq 1, length 64
```

Networking Commands: Login to remote host (ssh) *

- Protocol used to securely connect to a remote server/system
- It is very secure in the sense that it transfers the data in encrypted form between the host and the client
- It transfers inputs from the client to the host and relays back the output
- SSH runs at TCP/IP port 22

ssh user@10.



```
Network Working Group                                          T. Ylonen
Request for Comments: 4253          SSH Communications Security Corp
Category: Standards Track                                C. Lonvick, Ed.
                                                     Cisco Systems, Inc.
                                                            January 2006
```

The Secure Shell (SSH) Transport Layer Protocol

ssh ✓

Usage: **ssh <username>@<remote host>**

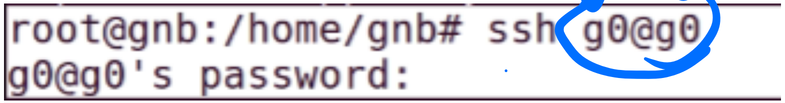
Common Options:

-p : Port no to connect to

-q : Supress all errors and warnings

- v : verbose mode (echo everything while establishing a connection)

```
#!/bin/bash
ssh cn@10.250.61.4
```



```
root@gnb:/home/gnb# ssh g0@g0
g0@g0's password:
```

Networking Commands: File transfer Operation (scp)

- Allows secure transfer of files and directories between the local host and the remote host or between two remote hosts
- Uses SSH for data transfer and uses the same authentication and provides the same security as SSH
- Known for its simplicity, security and pre-installed availability

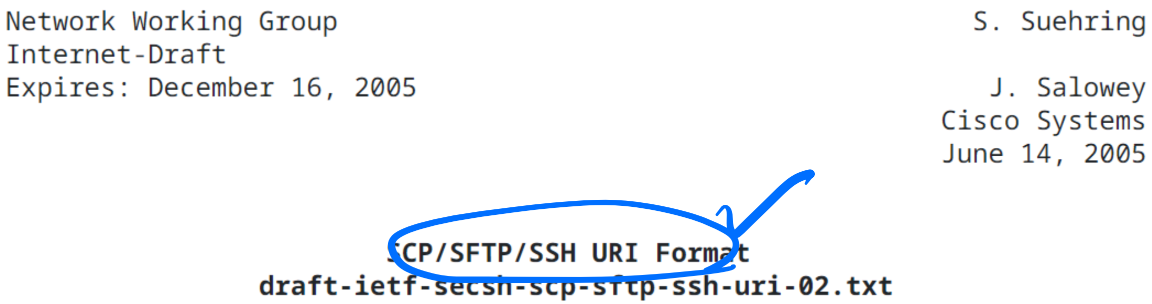
Usage: **scp <OPTIONS> [[user@]host1:]file1 … [[user@]host2:]file2**

Common Options:

-P: Port to connect on the remote host

-r : Recursively copy entire directories.

-q: Disable the progress meter.



```
Network Working Group                                        S. Suehring
Internet-Draft
Expires: December 16, 2005                                    J. Salowey
                                                           Cisco Systems
                                                           June 14, 2005

                        SCP/SFTP/SSH URI Format
                draft-ietf-secsh-scp-sftp-ssh-uri-02.txt
```

```
#!/bin/bash
scp $SSL_FOLDER cn@10.250.61.98:/home/cn/Documents
```

Networking Commands: File transfer Operation (rsync)

- Software utility for Unix-Like systems that efficiently sync files and directories between two hosts or machines
- Offers a large no of options that control every aspect of its behavior and permit very flexible specifications of the set of files to be copied.
- Famous for delta transfer algorithm by sending only the differences between the source files and existing files in the directory.

Usage: **rsync <option> <source> <destination>**

Common Options:

-a : archive mode

- r: recurse into subdirectories

- v: increase verbosity

- z: compress the data file during transfer

```
#!/bin/bash
rsync main.c user@remote-host:/home/cn/Desktop
```

thank you!

email:
k.kondepu@iitdh.ac.in