

LINUX ASSIGNMENT

NAME : VARSHA .K

CLASS : 3C

USN : ENG24CY0066

NICE - START PROCESS WITH MODIFIED PRIORITY

- **What is ‘nice’?**

Ans: the ‘nice’ command in linux is used to start a new process with a modified CPU scheduling priority

It assigns a numeric “niceness” value that influences how politely a process uses CPU time

The niceness value ranges from -20 which is highest priority to +19 which is lowest priority

- **Why is ‘nice’ used?**

Ans: the ‘nice’ command is used to control how aggressively or politely a process completes for CPU resources

It helps avoid system slowdowns , improves performances of important tasks, ensures fair multitasking and gives system administrations control over resource usage

- **How does ‘nice’ work ?**

Ans : when a command is executed using ‘nice’ , linux assigns the specified nice value before the process starts running , this value affects the scheduler’s decision on how much CPU time the process receives
Higher niceness assigns lower priority and lower niceness gives higher priority to a process

- **Commands and output**

Ans: example 1: start process with niceness+10

Command :

Nice-n 10 python3 [script.py](#)

In top output

NI=10

PR = 30

Example B :

Start a process with higher priority (-5):

Sudo nice -n-5 make

In top output :

NI=-5

PR = 15

Check priority :

Ps-1

Change priority of running process:

Sudo renice -n 5 -p 3450

- **Options for ‘nice command’**

-n <valu> : sets niceness value

–adjustment <value> : long form of -n

<command> : command to run with modified priority

- **HISTORY of the ‘nice’ command**

Ans: the ‘nice’ concept originated in linux version 4(1973)

It was designed so program could run “nicely” without dominating system resources

Later,support for negative nice values was added for higher priority tasks

It is now part of the POSFIX.1 standard and supported across linux ,BSD,MACOS and other unix-like systems

1. WHAT is **tc**?

tc (Traffic Control) is a Linux command used to configure, monitor, and manage network traffic on network interfaces.

It is part of the iproute2 package and allows administrators to control:

- Bandwidth allocation
- Packet delay
- Packet loss
- Queue disciplines (qdisc)
- Traffic shaping
- Traffic policing
- Class-based control

In short, `tc` manages how packets are queued, shaped, dropped, or delayed in the Linux kernel.

2. WHY is `tc` used?

Administrators use `tc` to control network performance for many reasons:

✓ Traffic Shaping

Limit bandwidth for certain applications (e.g., limit backup server to 5 Mbps).

✓ Reduce Network Congestion

Prevent a single user/process from consuming full network speed.

✓ Simulate Poor Network Conditions

Useful for testing:

- add delay

- add packet loss
- add jitter

Example: simulate 200ms latency for app testing.

✓ **Quality of Service (QoS)**

Give high priority to important traffic (VoIP, video calls) and low priority to bulk downloads.

✓ **Traffic Policing**

Drop packets exceeding a defined bandwidth limit.

✓ **Network Emulation (netem)**

Test mobile network conditions on local machines.

3. HOW does `tc` work?

`tc` manages traffic through three core components:

1. Qdisc (Queue discipline)

Determines how packets are queued and transmitted.

Examples:

- `pfifo` (simple queue)
- `fq_codel` (fair queueing)
- `htb` (hierarchical token bucket)
- `netem` (network emulator)

2. Classes

Inside a qdisc, traffic can be divided into multiple classes (with different priorities or bandwidth limits).

3. Filters

Specify *which traffic* goes into *which class* (based on IP, port, protocol, etc).

How it works step-by-step:

1. Attach a qdisc to a network interface.
2. Create classes inside the qdisc (optional).
3. Apply filters to direct packets to classes.
4. Configure bandwidth, priority, delay, loss, etc.

The Linux kernel then enforces these rules on all outgoing packets.

4. Commands With Expected Output

4.1 Show all traffic control settings

Command:

```
tc qdisc show
```

Output example:

```
qdisc fq_codel 0: dev eth0 root refcnt 2 limit 1000
```

4.2 Limit bandwidth to 1 Mbps using HTB

Commands:

```
tc qdisc add dev eth0 root handle 1: htb default 10
tc class add dev eth0 parent 1: classid 1:10 htb rate
1mbit ceil 1mbit
```

Output (check with):

```
tc -s qdisc show dev eth0
```

Sample output:

```
class htb 1:10 root rate 1Mbit ceil 1Mbit
Sent 210 packets, 31000 bytes
```

4.3 Add network delay (200ms)

Command:

```
tc qdisc add dev eth0 root netem delay 200ms
```

Output (verification):

```
tc qdisc show dev eth0
```

Output:

```
qdisc netem 10: dev eth0 root delay 200ms
```

4.4 Add packet loss (5%)

```
tc qdisc add dev wlan0 root netem loss 5%
```

4.5 Delete all rules

```
tc qdisc del dev eth0 root
```

5. Options / Flags for tc

Flag / Command	Meaning
qdisc	Manage queue disciplines
class	Create/manage classes
filter	Match traffic and attach to classes
add	Add qdisc/class/filter
del	Delete qdisc/class/filter
change	Modify existing rules
show or list	Show current configuration
-s	Show statistics (byte/packet counts)
dev <interface>	Specify network device (eth0, wlan0, lo)
rate	Set bandwidth limit
ceil	Maximum allowed bandwidth

delay	Add latency
loss	Add packet loss
jitter	Add variation in delay

6. History of the **tc** Command

- Introduced as part of the Linux Traffic Control subsystem in Linux Kernel 2.2 (1999).
- Originally created to replace older, limited QoS tools.
- Fully integrated with iproute2 suite, replacing old **ifconfig/route**.
- Gained support for netem (network emulator) in 2003 by Stephen Hemminger.
- Evolved to support:
 - HTB shaping
 - HFSC scheduling
 - fq_codel (default modern qdisc)
 - CAKE qdisc (for home routers)

Today, it is the standard Linux tool for professional and academic network shaping/testing.

7. Additional Technical Information

- **tc** only shapes outgoing traffic by default.
- Ingress policing requires a special *ingress qdisc*.
- Kernel handles packet scheduling using algorithms like:
 - **RED (Random Early Detection)**
 - **SFQ (Stochastic Fair Queueing)**
 - **fq_codel (reduces bufferbloat)**
- Commonly used in:
 - **Data centers**
 - **Routers**
 - **Cloud servers**
 - **Network testing labs**

For nice command :

```
varsha_k@varsha:~$ bash
varsha_k@varsha:~$ nice ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
varsha_k@varsha:~$ ps -l
F S  UID      PID  PPID C PRI  NI ADDR SZ WCHAN TTY          TIME CMD
0 S  1000     3067  3060  0  80   0 -  2755 do_wai pts/0    00:00:00 bash
0 S  1000     3107  3067  0  80   0 -  2756 do_wai pts/0    00:00:00 bash
0 S  1000     3142  3107  0  80   0 -  2756 do_wai pts/0    00:00:00 bash
0 S  1000     3272  3142  0  80   0 -  2756 do_wai pts/0    00:00:00 bash
0 R  1000     3355  3272  0  80   0 -  3445 -           pts/0    00:00:00 ps
varsha_k@varsha:~$
```

At least elaborate five to six Option and Flags related to the said command.
Give sources for detailed flags and options

For nice:

1. -n <value>

explanation

- -n 10 → increases the niceness (lower priority)
- Allowed range: **-20 to +19**
- Negative values need **sudo/root**

2. --help

Explanation

Shows a short reference guide for all flags.

--version

What it does

Shows version information of GNU Coreutils `nice`.

Usage

```
nice --version
```

4 Using `nice` WITHOUT flags

What it does

If you run:

```
nice command
```

It automatically runs the program with a default **increment** (usually +10).

Example

```
nice gcc file.c
```

This raises niceness by **10**, making it lower priority.

5 Negative Niceness (requires sudo)

What it does

Increases priority (reduces niceness).

Usage

```
sudo nice -n -5 ./server
```

Explanation

- Negative = higher priority
- Only root can reduce niceness

Reliable Sources for Detailed Flags & Options

1. GNU Coreutils Official Manual

https://www.gnu.org/software/coreutils/manual/html_node/nice-invocation.html

2. Linux Man Pages (man7.org)

<https://man7.org/linux/man-pages/man1/nice.1.html>

3. TLDP (The Linux Documentation Project)

<https://tldp.org/LDP/abs/html/process-control.html>

4. Ubuntu Manpages

<https://manpages.ubuntu.com/manpages/noble/man1/nice.1.html>