NAME

top − display Linux processes

SYNOPSIS

top −**hv**|−**bcEeHiOSs1** −**d** secs −**n** max −**u**|**U** user −**p** pids −**o** field −**w** [cols]

The traditional switches '−' and whitespace are optional.

DESCRIPTION

The **top** program provides a dynamic real-time view of a running system. It can display **system** summary information as well as a list of **processes** or **threads** currently being managed by the Linux kernel. The types of system summary information shown and the types, order and size of information displayed for processes are all user configurable and that configuration can be made persistent across restarts.

The program provides a limited interactive interface for process manipulation as well as a much more extensive interface for personal configuration −− encompassing every aspect of its operation. And while **top** is referred to throughout this document, you are free to name the program anything you wish. That new name, possibly an alias, will then be reflected on top's display and used when reading and writing a configuration file.

OVERVIEW

Documentation

The remaining Table of Contents

Operation

When operating top, the two most important keys are the help (h or ?) key and quit ('q') key. Alternatively, you could simply use the traditional interrupt key (^C) when you're done.

When started for the first time, you'll be presented with these traditional elements on the main top screen: 1) Summary Area; 2) Fields/Columns Header; 3) Task Area. Each of these will be explored in the sections that follow. There is also an Input/Message line between the Summary Area and Columns Header which needs no further explanation.

The main top screen is *generally* quite adaptive to changes in terminal dimensions under X-Windows. Other top screens may be less so, especially those with static text. It ultimately depends, however, on your particular window manager and terminal emulator. There may be occasions when their view of terminal size and current contents differs from top's view, which is always based on operating system calls.

Following any re-size operation, if a top screen is corrupted, appears incomplete or disordered, simply typing something innocuous like a punctuation character or cursor motion key will usually restore it. In extreme cases, the following sequence almost certainly will:

key/cmd	*objective*
^Z	**suspend** top
fg	**resume** top
<Left>	force a screen**r edraw** (if necessary)

But if the display is still corrupted, there is one more step you could try. Insert this command after top has been suspended but before resuming it.

key/cmd	*objective*
reset	restore your**terminal settings**

Note: the width of top's display will be limited to 512 positions. Displaying all fields requires approximately 250 characters. Remaining screen width is usually allocated to any variable width columns currently visible. The variable width columns, such as COMMAND, are noted in topic 3a. DESCRIPTIONS of Fields. Actual output width may also be influenced by the −w switch, which is discussed in topic 1. COMMAND−LINE Options.

Lastly, some of top's screens or functions require the use of cursor motion keys like the standard arrow keys plus the Home, End, PgUp and PgDn keys. If your terminal or emulator does not provide those keys, the following combinations are accepted as alternatives:

key	*equivalent-keys*
Left	alt +**h**
Down	alt +**j**
Up	alt +**k**
Right	alt +**l**
Home	alt + ctrl +**h**
PgDn	alt + ctrl +**j**
PgUp	alt + ctrl +**k**
End	alt + ctrl +**l**

The **Up** and **Down** arrow keys have special significance when prompted for line input terminated with the <Enter> key. Those keys, or their aliases, can be used to retrieve previous input lines which can then be edited and re-input. And there are four additional keys available with line oriented input.

key	*special-significance*
Up	recall**older** strings for re-editing
Down	recall**newer** strings or **erase** entire line
Insert	toggle between**insert** and **o vertype** modes
Delete	character**r emoved** at cursor, moving others left
Home	jump to**beginning** of input line
End	jump to**end** of input line

Linux Memory Types

For our purposes there are three types of memory, and one is optional. First is physical memory, a limited resource where code and data must reside when executed or referenced. Next is the optional swap file, where modified (dirty) memory can be saved and later retrieved if too many demands are made on physical memory. Lastly we have virtual memory, a nearly unlimited resource serving the following goals:

1. abstraction, free from physical memory addresses/limits
2. isolation, every process in a separate address space
3. sharing, a single mapping can serve multiple needs
4. flexibility, assign a virtual address to a file

Regardless of which of these forms memory may take, all are managed as pages (typically 4096 bytes) but expressed by default in top as KiB (kibibyte). The memory discussed under topic ‘2c. MEMORY Usage’ deals with physical memory and the swap file for the system as a whole. The memory reviewed in topic ‘3. FIELDS / Columns Display’ embraces all three memory types, but for individual processes.

For each such process, every memory page is restricted to a single quadrant from the table below. Both physical memory and virtual memory can include any of the four, while the swap file only includes #1 through #3. The memory in quadrant #4, when modified, acts as its own dedicated swap file.

	Private	**Shared**
	1	**2**
Anonymous	. stack . malloc() . brk()/sbrk() . mmap(PRIVATE, ANON)	. POSIX shm* . mmap(SHARED, ANON)
File-backed	. pgms/shared libs . mmap(PRIVATE, fd)	. mmap(SHARED, fd)
	3	**4**

The following may help in interpreting process level memory values displayed as scalable columns and discussed under topic ‘3a. DESCRIPTIONS of Fields’.

%MEM − simply RES divided by total physical memory
CODE − the ‘pgms’ portion of quadrant **3**
DATA − the entire quadrant **1** portion of VIRT plus all
explicit mmap file-backed pages of quadrant **3**
RES − anything occupying physical memory which, beginning with
Linux-4.5, is the sum of the following three fields:
RSan − quadrant **1** pages, which include any
former quadrant **3** pages if modified
RSfd − quadrant **3** and quadrant **4** pages

RSsh − quadrant **2** pages
RSlk − subset of RES which cannot be swapped out (any quadrant)
SHR − subset of RES (excludes **1**, includes all **2** & **4**, some **3**)
SWAP − potentially any quadrant except **4**
USED − simply the sum of RES and SWAP
VIRT − everything in-use and/or reserved (all quadrants)

Note: Even though program images and shared libraries are considered *private* to a process, they will be accounted for as *shared* (SHR) by the kernel.

1. COMMAND-LINE Options

The command-line syntax for top consists of:

−**hv**|−**bcEeHiOSs1** −**d** secs −**n** max −**u**|**U** user −**p** pids −**o** field −**w** [cols]

The typically mandatory switch (‘−’) and even whitespace are completely optional.

−**h** | −**v** :*Help/Version*
Show library version and the usage prompt, then quit.

−**b** :*Batch-mode* operation
Starts top in Batch mode, which could be useful for sending output from top to other programs or to a file. In this mode, top will not accept input and runs until the iterations limit you’ve set with the ‘−n’ command−line option or until killed.

−**c** :*Command-line/Program-name* toggle
Starts top with the last remembered ‘c’ state reversed. Thus, if top was displaying command lines, now that field will show program names, and vice versa. See the ‘c’ interactive command for additional information.

−**d** :*Delay-time* interval as: **-d ss.t** (*secs.tenths*)
Specifies the delay between screen updates, and overrides the corresponding value in one’s personal configuration file or the startup default. Later this can be changed with the ‘d’ or ‘s’ interactive commands.

Fractional seconds are honored, but a negative number is not allowed. In all cases, however, such changes are prohibited if top is running in Secure mode, except for root (unless the ‘s’ command−line option was used). For additional information on Secure mode see topic 6d. SYSTEM Restrictions File.

−**e** :*Enforce-Task-Memory-Scaling* as: **-e k**| **m** | **g** | **t** | **p**
Instructs top to force task area memory to be scaled as:
k − kibibytes
m − mebibytes
g − gibibytes
t − tebibytes
p − pebibytes

Later this can be changed with the ‘e’ command toggle.

−**E** :*Enforce-Summary-Memory-Scaling* as: **-E k| m | g | t | p | e**
Instructs top to force summary area memory to be scaled as:
k − kibibytes
m − mebibytes
g − gibibytes
t − tebibytes
p − pebibytes
e − exbibytes

Later this can be changed with the ‘E’ command toggle.

−**H** :*Threads-mode* operation
Instructs top to display individual threads. Without this command−line option a summation of all threads in each process is shown. Later this can be changed with the ‘H’ interactive command.

−**i** :*Idle-process* toggle
Starts top with the last remembered ‘i’ state reversed. When this toggle is*Of f*, tasks that have not used any CPU since the last update will not be displayed. For additional information regarding this toggle see topic 4c. TASK AREA Commands, SIZE.

−**n** :*Number-of-iterations* limit as: **−n number**
Specifies the maximum number of iterations, or frames, top should produce before ending.

−**o** :*Override-sort-field* as: **−o fieldname**
Specifies the name of the field on which tasks will be sorted, independent of what is reflected in the configuration file. You can prepend a ‘+’ or ‘−’ to the field name to also override the sort direction. A leading ‘+’ will force sorting high to low, whereas a ‘−’ will ensure a low to high ordering.

This option exists primarily to support automated/scripted batch mode operation.

−**O** :*Output-field-names*
This option acts as a form of help for the above −o option. It will cause top to print each of the available field names on a separate line, then quit. Such names are subject to NLS (National Language Support) translation.

−**p** :*Monitor-PIDs* mode as: **−pN1 -pN2 ...** or **−pN1,N2,N3 ...**
Monitor only processes with specified process IDs. This option can be given up to 20 times, or you can provide a comma delimited list with up to 20 pids. Co-mingling both approaches is permitted.

A pid value of zero will be treated as the process id of the top program itself once it is running.

This is a command−line option only and should you wish to return to normal operation, it is not necessary to quit and restart top −− just issue any of these interactive commands: ‘=’, ‘u’ or ‘U’.

The ‘p’, ‘u’ and ‘U’ command−line options are mutually exclusive.

−**s** :*Secure-mode* operation
Starts top with secure mode forced, even for root. This mode is far better controlled through a system configuration file (see topic 6. FILES).

−**S** :*Cumulative-time* toggle
Starts top with the last remembered ‘S’ state reversed. When Cumulative time mode is *On*, each process is listed with the cpu time that it and its dead children have used. See the ‘S’ interactive command for additional information regarding this mode.

−**u** | −**U** :*User-filter-mode* as: **-u** | **-U number** or **name**
Display only processes with a user id or user name matching that given. The ‘−u’ option matches on *effective* user whereas the ‘−U’ option matches on *any* user (real, effective, saved, or filesystem).

Prepending an exclamation point (‘!’) to the user id or name instructs top to display only processes with users not matching the one provided.

The ‘p’, ‘u’ and ‘U’ command−line options are mutually exclusive.

−**w** :*Output-width-override* as: **-w** [**number**]
In Batch mode, when used without an argument top will format output using the COLUMNS= and LINES= environment variables, if set. Otherwise, width will be fixed at the maximum 512 columns. With an argument, output width can be decreased or increased (up to 512) but the number of rows is considered unlimited.

In normal display mode, when used without an argument top will *attempt* to format output using the COLUMNS= and LINES= environment variables, if set. With an argument, output width can only be decreased, not increased. Whether using environment variables or an argument with −w, when *not* in Batch mode actual terminal dimensions can never be exceeded.

Note: Without the use of this command−line option, output width is always based on the terminal at which top was invoked whether or not in Batch mode.

−**1** :*Single/Separate-Cpu-States* toggle
Starts top with the last remembered Cpu States portion of the summary area reversed. Either all cpu information will be displayed in a single line or each cpu will be displayed separately, depending on the state of the NUMA Node command toggle (’2’).

See the ‘1’ and ’2’ interactive commands for additional information.

2. SUMMARY Display

Each of the following three areas are individually controlled through one or more interactive commands. See topic 4b. SUMMARY AREA Commands for additional information regarding these provisions.

2a. UPTIME and LOAD Averages

This portion consists of a single line containing:
program or **window** name, depending on display mode
current time and length of time since last boot
total number of users
system load avg over the last 1, 5 and 15 minutes

2b. TASK and CPU States

This portion consists of a minimum of two lines. In an SMP environment, additional lines can reflect individual CPU state percentages.

Line 1 shows total **tasks** or **threads**, depending on the state of the Threads-mode toggle. That total is further classified as:

running; sleeping; stopped; zombie

Line 2 shows CPU state percentages based on the interval since the last refresh.

As a default, percentages for these individual categories are displayed. Where two labels are shown below, those for more recent kernel versions are shown first.
us, **user** : time running un-niced user processes
sy, **system** : time running kernel processes
ni, **nice** : time running niced user processes
id, **idle** : time spent in the kernel idle handler
wa, **IO-wait** : time waiting for I/O completion
hi : time spent servicing hardware interrupts
si : time spent servicing software interrupts
st : time stolen from this vm by the hypervisor

In the alternate cpu states display modes, beyond the first tasks/threads line, an abbreviated summary is shown consisting of these elements:

```
        a   b    c   d
%Cpu(s):  75.0/25.0  100[ ...
```

Where: a) is the ‘user’ (us + ni) percentage; b) is the ‘system’ (sy + hi + si) percentage; c) is the total; and d) is one of two visual graphs of those representations. See topic 4b. SUMMARY AREA Commands and the ‘t’ command for additional information on that special 4-way toggle.

2c. MEMORY Usage

This portion consists of two lines which may express values in kibibytes (KiB) through exbibytes (EiB) depending on the scaling factor enforced with the ‘E’ interactive command.

As a default, Line 1 reflects physical memory, classified as:
total, free, used and buff/cache

Line 2 reflects mostly virtual memory, classified as:
total, free, used and avail (which is physical memory)

The **avail** number on line 2 is an estimation of physical memory available for starting new applications, without swapping. Unlike the **free** field, it attempts to account for readily reclaimable page cache and memory slabs. It is available on kernels 3.14, emulated on kernels 2.6.27+, otherwise the same as **free**.

In the alternate memory display modes, two abbreviated summary lines are shown consisting of these elements:

```
        a   b        c
GiB Mem : 18.7/15.738   [ ...
GiB Swap:  0.0/7.999    [ ...
```

Where: a) is the percentage used; b) is the total available; and c) is one of two visual graphs of those representations.

In the case of physical memory, the percentage represents the **total** minus the estimated **avail** noted above. The ‘Mem’ graph itself is divided between **used** and any remaining memory not otherwise accounted for by **avail**. See topic 4b. SUMMARY AREA Commands and the ‘m’ command for additional information on that special 4-way toggle.

This table may help in interpreting the scaled values displayed:

KiB = kibibyte = 1024 bytes
MiB = mebibyte = 1024 KiB = 1,048,576 bytes
GiB = gibibyte = 1024 MiB = 1,073,741,824 bytes
TiB = tebibyte = 1024 GiB = 1,099,511,627,776 bytes
PiB = pebibyte = 1024 TiB = 1,125,899,906,842,624 bytes
EiB = exbibyte = 1024 PiB = 1,152,921,504,606,846,976 bytes

3. FIELDS / Columns

3a. DESCRIPTIONS of Fields

Listed below are top's available process fields (columns). They are shown in strict ascii alphabetical order. You may customize their position and whether or not they are displayable with the 'f' or 'F' (Fields Management) interactive commands.

Any field is selectable as the sort field, and you control whether they are sorted high-to-low or low-to-high. For additional information on sort provisions see topic 4c. TASK AREA Commands, SORTING.

The fields related to physical memory or virtual memory reference '(KiB)' which is the unsuffixed display mode. Such fields may, however, be scaled from KiB through PiB. That scaling is influenced via the 'e' interactive command or established for startup through a build option.

1. **%CPU** −− CPU Usage
   The task's share of the elapsed CPU time since the last screen update, expressed as a percentage of total CPU time.

   In a true SMP environment, if a process is multi-threaded and top is *not* operating in Threads mode, amounts greater than 100% may be reported. You toggle Threads mode with the 'H' interactive command.

   Also for multi-processor environments, if Irix mode is *Off*, top will operate in Solaris mode where a task's cpu usage will be divided by the total number of CPUs. You toggle Irix/Solaris modes with the 'I' interactive command.

   Note: When running in forest view mode ('V') with children collapsed ('v'), this field will also include the CPU time of those unseen children. See topic 4c. TASK AREA Commands, CONTENT for more information regarding the 'V' and 'v' toggles.

2. **%MEM** −− Memory Usage (RES)
   A task's currently resident share of available physical memory.

   See 'OVERVIEW, Linux Memory Types' for additional details.

3. **CGNAME** −− Control Group Name
   The name of the control group to which a process belongs, or '−' if not applicable for that process.

   This will typically be the last entry in the full list of control groups as shown under the next heading (CGROUPS). And as is true there, this field is also variable width.

4. **CGROUPS** −− Control Groups
   The names of the control group(s) to which a process belongs, or '−' if not applicable for that process.

   Control Groups provide for allocating resources (cpu, memory, network bandwidth, etc.) among installation-defined groups of processes. They enable fine-grained control over allocating, denying,

prioritizing, managing and monitoring those resources.

Many different hierarchies of cgroups can exist simultaneously on a system and each hierarchy is attached to one or more subsystems. A subsystem represents a single resource.

Note: The CGROUPS field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

5. **CODE** −− Code Size (KiB)
The amount of physical memory currently devoted to executable code, also known as the Text Resident Set size or TRS.

See 'OVERVIEW, Linux Memory Types' for additional details.

6. **COMMAND** −− Command **Name** or Command **Line**
Display the command line used to start a task or the name of the associated program. You toggle between command *line* and *name* with 'c', which is both a command−line option and an interactive command.

When you've chosen to display command lines, processes without a command line (like kernel threads) will be shown with only the program name in brackets, as in this example:
[kthreadd]

This field may also be impacted by the forest view display mode. See the 'V' interactive command for additional information regarding that mode.

Note: The COMMAND field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. This is especially true for this field when command lines are being displayed (the 'c' interactive command.) See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

7. **DATA** −− Data + Stack Size (KiB)
The amount of private memory *reserved* by a process. It is also known as the Data Resident Set or DRS. Such memory may not yet be mapped to physical memory (RES) but will always be included in the virtual memory (VIRT) amount.

See 'OVERVIEW, Linux Memory Types' for additional details.

8. **ENVIRON** −− Environment variables
Display all of the environment variables, if any, as seen by the respective processes. These variables will be displayed in their raw native order, not the sorted order you are accustomed to seeing with an unqualified 'set'.

Note: The ENVIRON field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. This is especially true for this field. See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

9. **Flags** −− Task Flags
   This column represents the task's current scheduling flags which are expressed in hexadecimal notation and with zeros suppressed. These flags are officially documented in <linux/sched.h>.

10. **GID** −− Group Id
    The *effective* group ID.

11. **GROUP** −− Group Name
    The *effective* group name.

12. **LXC** −− Lxc Container Name
    The name of the lxc container within which a task is running. If a process is not running inside a container, a dash ('−') will be shown.

13. **NI** −− Nice Value
    The nice value of the task. A negative nice value means higher priority, whereas a positive nice value means lower priority. Zero in this field simply means priority will not be adjusted in determining a task's dispatch-ability.

14. **NU** −− Last known NUMA node
    A number representing the NUMA node associated with the last used processor ('P'). When -1 is displayed it means that NUMA information is not available.

    See the ''2' and '3' interactive commands for additional NUMA provisions affecting the summary area.

15. **OOMa** −− Out of Memory Adjustment Factor
    The value, ranging from -1000 to +1000, added to the current out of memory score (OOMs) which is then used to determine which task to kill when memory is exhausted.

16. **OOMs** −− Out of Memory Score
    The value, ranging from 0 to +1000, used to select task(s) to kill when memory is exhausted. Zero translates to 'never kill' whereas 1000 means 'always kill'.

17. **P** −− Last used CPU (SMP)
    A number representing the last used processor. In a true SMP environment this will likely change frequently since the kernel intentionally uses weak affinity. Also, the very act of running top may break this weak affinity and cause more processes to change CPUs more often (because of the extra demand for cpu time).

18. **PGRP** −− Process Group Id
    Every process is member of a unique process group which is used for distribution of signals and by terminals to arbitrate requests for their input and output. When a process is created (forked), it becomes a member of the process group of its parent. By convention, this value equals the process ID (see PID) of the first member of a process group, called the process group leader.

19. **PID** −− Process Id
    The task's unique process ID, which periodically wraps, though never restarting at zero. In kernel terms, it is a dispatchable entity defined by a task_struct.

This value may also be used as: a process group ID (see PGRP); a session ID for the session leader (see SID); a thread group ID for the thread group leader (see TGID); and a TTY process group ID for the process group leader (see TPGID).

20. **PPID** −− Parent Process Id
    The process ID (pid) of a task's parent.

21. **PR** −− Priority
    The scheduling priority of the task. If you see 'rt' in this field, it means the task is running under real time scheduling priority.

    Under linux, real time priority is somewhat misleading since traditionally the operating itself was not preemptible. And while the 2.6 kernel can be made mostly preemptible, it is not always so.

22. **RES** −− Resident Memory Size (KiB)
    A subset of the virtual address space (VIRT) representing the non-swapped physical memory a task is currently using. It is also the sum of the RSan, RSfd and RSsh fields.

    It can include private anonymous pages, private pages mapped to files (including program images and shared libraries) plus shared anonymous pages. All such memory is backed by the swap file represented separately under SWAP.

    Lastly, this field may also include shared file-backed pages which, when modified, act as a dedicated swap file and thus will never impact SWAP.

    See 'OVERVIEW, Linux Memory Types' for additional details.

23. **RSan** −− Resident Anonymous Memory Size (KiB)
    A subset of resident memory (RES) representing private pages not mapped to a file.

24. **RSfd** −− Resident File-Backed Memory Size (KiB)
    A subset of resident memory (RES) representing the implicitly shared pages supporting program images and shared libraries. It also includes explicit file mappings, both private and shared.

25. **RSlk** −− Resident Locked Memory Size (KiB)
    A subset of resident memory (RES) which cannot be swapped out.

26. **RSsh** −− Resident Shared Memory Size (KiB)
    A subset of resident memory (RES) representing the explicitly shared anonymous shm*/mmap pages.

27. **RUID** −− Real User Id
    The *real* user ID.

28. **RUSER** −− Real User Name
    The *real* user name.

29. **S** −− Process Status
    The status of the task which can be one of:
    D = uninterruptible sleep
    I = idle

R = running
S = sleeping
T = stopped by job control signal
t = stopped by debugger during trace
Z = zombie

Tasks shown as running should be more properly thought of as ready to run -- their task_struct is simply represented on the Linux run-queue. Even without a true SMP machine, you may see numerous tasks in this state depending on top's delay interval and nice value.

30. **SHR** -- Shared Memory Size (KiB)
A subset of resident memory (RES) that may be used by other processes. It will include shared anonymous pages and shared file-backed pages. It also includes private pages mapped to files representing program images and shared libraries.

See 'OVERVIEW, Linux Memory Types' for additional details.

31. **SID** -- Session Id
A session is a collection of process groups (see PGRP), usually established by the login shell. A newly forked process joins the session of its creator. By convention, this value equals the process ID (see PID) of the first member of the session, called the session leader, which is usually the login shell.

32. **SUID** -- Saved User Id
The *saved* user ID.

33. **SUPGIDS** -- Supplementary Group IDs
The IDs of any supplementary group(s) established at login or inherited from a task's parent. They are displayed in a comma delimited list.

Note: The SUPGIDS field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

34. **SUPGRPS** -- Supplementary Group Names
The names of any supplementary group(s) established at login or inherited from a task's parent. They are displayed in a comma delimited list.

Note: The SUPGRPS field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

35. **SUSER** -- Saved User Name
The *saved* user name.

36. **SWAP** -- Swapped Size (KiB)
The formerly resident portion of a task's address space written to the swap file when physical memory becomes over committed.

See 'OVERVIEW, Linux Memory Types' for additional details.

37. **TGID** −− Thread Group Id
    The ID of the thread group to which a task belongs. It is the PID of the thread group leader. In kernel terms, it represents those tasks that share an mm_struct.

38. **TIME** −− CPU Time
    Total CPU time the task has used since it started. When Cumulative mode is *On*, each process is listed with the cpu time that it and its dead children have used. You toggle Cumulative mode with 'S', which is both a command−line option and an interactive command. See the 'S' interactive command for additional information regarding this mode.

39. **TIME+** −− CPU Time, hundredths
    The same as TIME, but reflecting more granularity through hundredths of a second.

40. **TPGID** −− Tty Process Group Id
    The process group ID of the foreground process for the connected tty, or −1 if a process is not connected to a terminal. By convention, this value equals the process ID (see PID) of the process group leader (see PGRP).

41. **TTY** −− Controlling Tty
    The name of the controlling terminal. This is usually the device (serial port, pty, etc.) from which the process was started, and which it uses for input or output. However, a task need not be associated with a terminal, in which case you'll see '?' displayed.

42. **UID** −− User Id
    The *effective* user ID of the task's owner.

43. **USED** −− Memory in Use (KiB)
    This field represents the non-swapped physical memory a task is using (RES) plus the swapped out portion of its address space (SWAP).

    See 'OVERVIEW, Linux Memory Types' for additional details.

44. **USER** −− User Name
    The *effective* user name of the task's owner.

45. **VIRT** −− Virtual Memory Size (KiB)
    The total amount of virtual memory used by the task. It includes all code, data and shared libraries plus pages that have been swapped out and pages that have been mapped but not used.

    See 'OVERVIEW, Linux Memory Types' for additional details.

46. **WCHAN** −− Sleeping in Function
    This field will show the name of the kernel function in which the task is currently sleeping. Running tasks will display a dash ('−') in this column.

47. **nDRT** −− Dirty Pages Count
    The number of pages that have been modified since they were last written to auxiliary storage. Dirty pages must be written to auxiliary storage before the corresponding physical memory location can be used for some other virtual page.

This field was deprecated with linux 2.6 and is always zero.

48. **nMaj** −− Major Page Fault Count
    The number of **major** page faults that have occurred for a task. A page fault occurs when a process attempts to read from or write to a virtual page that is not currently present in its address space. A major page fault is when auxiliary storage access is involved in making that page available.

49. **nMin** −− Minor Page Fault count
    The number of **minor** page faults that have occurred for a task. A page fault occurs when a process attempts to read from or write to a virtual page that is not currently present in its address space. A minor page fault does not involve auxiliary storage access in making that page available.

50. **nTH** −− Number of Threads
    The number of threads associated with a process.

51. **nsIPC** −− IPC namespace
    The Inode of the namespace used to isolate interprocess communication (IPC) resources such as System V IPC objects and POSIX message queues.

52. **nsMNT** −− MNT namespace
    The Inode of the namespace used to isolate filesystem mount points thus offering different views of the filesystem hierarchy.

53. **nsNET** −− NET namespace
    The Inode of the namespace used to isolate resources such as network devices, IP addresses, IP routing, port numbers, etc.

54. **nsPID** −− PID namespace
    The Inode of the namespace used to isolate process ID numbers meaning they need not remain unique. Thus, each such namespace could have its own ‘init/systemd’ (PID #1) to manage various initialization tasks and reap orphaned child processes.

55. **nsUSER** −− USER namespace
    The Inode of the namespace used to isolate the user and group ID numbers. Thus, a process could have a normal unprivileged user ID outside a user namespace while having a user ID of 0, with full root privileges, inside that namespace.

56. **nsUTS** −− UTS namespace
    The Inode of the namespace used to isolate hostname and NIS domain name. UTS simply means "UNIX Time-sharing System".

57. **vMj** −− Major Page Fault Count Delta
    The number of **major** page faults that have occurred since the last update (see nMaj).

58. **vMn** −− Minor Page Fault Count Delta
    The number of **minor** page faults that have occurred since the last update (see nMin).

3b. MANAGING Fields

After pressing the interactive command ‘f’ or ‘F’ (Fields Management) you will be presented with a screen showing: 1) the ‘current’ window name; 2) the designated sort field; 3) all fields in their current order along

with descriptions. Entries marked with an asterisk are the currently displayed fields, screen width permitting.

- As the on screen instructions indicate, you navigate among the fields with the **Up** and **Down** arrow keys. The PgUp, PgDn, Home and End keys can also be used to quickly reach the first or last available field.

- The **Right** arrow key selects a field for repositioning and the **Left** arrow key or the <**Enter**> key commits that field's placement.

- The '**d**' key or the <**Space**> bar toggles a field's display status, and thus the presence or absence of the asterisk.

- The '**s**' key designates a field as the sort field. See topic 4c. TASK AREA Commands, SORTING for additional information regarding your selection of a sort field.

- The '**a**' and '**w**' keys can be used to cycle through all available windows and the '**q**' or <**Esc**> keys exit Fields Management.

The Fields Management screen can also be used to change the 'current' window/field group in either full−screen mode or alternate−display mode. Whatever was targeted when 'q' or <Esc> was pressed will be made current as you return to the top display. See topic 5. ALTERNATE−DISPLAY Provisions and the 'g' interactive command for insight into 'current' windows and field groups.

Note: Any window that has been scrolled *horizontally* will be reset if any field changes are made via the Fields Management screen. Any *vertical* scrolled position, however, will not be affected. See topic 5c. SCROLLING a Window for additional information regarding vertical and horizontal scrolling.

4. INTERACTIVE Commands

Listed below is a brief index of commands within categories. Some commands appear more than once −− their meaning or scope may vary depending on the context in which they are issued.

4a. *Global-Commands*
<Ent/Sp> ?, =, 0,
A, B, d, E, e, g, h, H, I, k, q, r, s, W, X, Y, Z
4b. *Summary-Area-Commands*
C, l, t, m, 1, 2, 3, 4, !
4c. *Task-Area-Commands*
Appearance: b, J, j, x, y, z
Content: c, f, F, o, O, S, u, U, V, v
Size: #, i, n
Sorting: <, >, f, F, R
4d. *Color-Mapping*
<Ret>, a, B, b, H, M, q, S, T, w, z, 0 − 7
5b. *Commands-for-Windows*
−, _, =, +, A, a, g, G, w
5c. *Scrolling-a-Window*
C, Up, Dn, Left, Right, PgUp, PgDn, Home, End
5d. *Searching-in-a-Window*
L, &

4a. GLOBAL Commands

The global interactive commands are **always** available in both full−screen mode and alternate−display mode. However, some of these interactive commands are **not available** when running in Secure mode.

If you wish to know in advance whether or not your top has been secured, simply ask for help and view the system summary on the second line.

<**Enter**> or <**Space**> :*Refresh-Display*

These commands awaken top and following receipt of any input the entire display will be repainted. They also force an update of any hotplugged cpu or physical memory changes.

Use either of these keys if you have a large delay interval and wish to see current status,

? | **h** :*Help*

There are two help levels available. The first will provide a reminder of all the basic interactive commands. If top is*secur ed*, that screen will be abbreviated.

Typing ‘h’ or ‘?’ on that help screen will take you to help for those interactive commands applicable to alternate−display mode.

= :*Exit-Display-Limits*

Removes restrictions on what is shown. This command will reverse any ‘i’ (idle tasks), ‘n’ (max tasks) and ‘v’ (hide children) commands that might be active. It also provides for an exit from PID monitoring, User filtering, Other filtering, Locate processing and Combine Cpus mode.

Additionally, if the window has been scrolled it will be reset with this command.

0 :*Zero-Suppress* toggle

This command determines whether zeros are shown or suppressed for many of the fields in a task window. Fields like UID, GID, NI, PR or P are not affected by this toggle.

A :*Alternate-Display-Mode* toggle

This command will switch between full−screen mode and alternate−display mode. See topic 5. ALTERNATE−DISPLAY Provisions and the ‘g’ interactive command for insight into ‘current’ windows and field groups.

B :*Bold-Disable/Enable* toggle

This command will influence use of the bold terminfo capability and alters **both** the summary area and task area for the ‘current’ window. While it is intended primarily for use with dumb terminals, it can be applied anytime.

Note: When this toggle is *On* and top is operating in monochrome mode, the **entire display** will appear as normal text. Thus, unless the ‘x’ and/or ‘y’ toggles are using reverse for emphasis, there will be no visual confirmation that they are even on.

* **d** | **s** :*Change-Delay-Time-interval*

You will be prompted to enter the delay time, in seconds, between display updates.

Fractional seconds are honored, but a negative number is not allowed. Entering 0 causes (nearly) continuous updates, with an unsatisfactory display as the system and tty driver try to keep up with top’s demands. The delay value is inversely proportional to system loading, so set it with care.

If at any time you wish to know the current delay time, simply ask for help and view the system summary on the second line.

E :*Enforce-Summary-Memory-Scale* in Summary Area
With this command you can cycle through the available summary area memory scaling which ranges from KiB (kibibytes or 1,024 bytes) through EiB (exbibytes or 1,152,921,504,606,846,976 bytes).

If you see a '+' between a displayed number and the following label, it means that top was forced to truncate some portion of that number. By raising the scaling factor, such truncation can be avoided.

e :*Enforce-Task-Memory-Scale* in Task Area
With this command you can cycle through the available task area memory scaling which ranges from KiB (kibibytes or 1,024 bytes) through PiB (pebibytes or 1,125,899,906,842,624 bytes).

While top will try to honor the selected target range, additional scaling might still be necessary in order to accommodate current values. If you wish to see a more homogeneous result in the memory columns, raising the scaling range will usually accomplish that goal. Raising it too high, however, is likely to produce an all zero result which cannot be suppressed with the '0' interactive command.

g :*Choose-Another-Window/Field-Group*
You will be prompted to enter a number between 1 and 4 designating the field group which should be made the 'current' window. You will soon grow comfortable with these 4 windows, especially after experimenting with alternate−display mode.

H :*Threads-mode* toggle
When this toggle is *On*, individual threads will be displayed for all processes in all visible task windows. Otherwise, top displays a summation of all threads in each process.

I :*Irix/Solaris-Mode* toggle
When operating in Solaris mode ('I' toggled *Off*), a task's cpu usage will be divided by the total number of CPUs. After issuing this command, you'll be told the new state of this toggle.

* **k** :*Kill-a-task*
You will be prompted for a PID and then the signal to send.

Entering no PID or a negative number will be interpreted as the default shown in the prompt (the first task displayed). A PID value of zero means the top program itself.

The default signal, as reflected in the prompt, is SIGTERM. However, you can send any signal, via number or name.

If you wish to abort the kill process, do one of the following depending on your progress:
1) at the pid prompt, type an invalid number
2) at the signal prompt, type 0 (or any invalid signal)
3) at any prompt, type <Esc>

q :*Quit*

* **r** :*Renice-a-Task*
You will be prompted for a PID and then the value to nice it to.

Entering no PID or a negative number will be interpreted as the default shown in the prompt (the first task displayed). A PID value of zero means the top program itself.

A positive nice value will cause a process to lose priority. Conversely, a negative nice value will cause a process to be viewed more favorably by the kernel. As a general rule, ordinary users can only increase the nice value and are prevented from lowering it.

If you wish to abort the renice process, do one of the following depending on your progress:
1) at the pid prompt, type an invalid number
2) at the nice prompt, type <Enter> with no input
3) at any prompt, type <Esc>

W :*Write-the-Configuration-File*
This will save all of your options and toggles plus the current display mode and delay time. By issuing this command just before quitting top, you will be able restart later in exactly that same state.

X :*Extra-Fixed-Width*
Some fields are fixed width and not scalable. As such, they are subject to truncation which would be indicated by a '+' in the last position.

This interactive command can be used to alter the widths of the following fields:

field	*default*	*field*	*default*	*field*	*default*
GID	5	GROUP	8	WCHAN	10
RUID	5	LXC	8	nsIPC	10
SUID	5	RUSER	8	nsMNT	10
UID	5	SUSER	8	nsNET	10
		TTY	8	nsPID	10
		USER	8	nsUSER	10
				nsUTS	10

You will be prompted for the amount to be added to the default widths shown above. Entering zero forces a return to those defaults.

If you enter a negative number, top will automatically increase the column size as needed until there is no more truncated data. You can accelerate this process by reducing the delay interval or holding down the <Space> bar.

Note: Whether explicitly or automatically increased, the widths for these fields are never decreased by top. To narrow them you must specify a smaller number or restore the defaults.

Y :*Inspect-Other-Output*
After issuing the 'Y' interactive command, you will be prompted for a target PID. Typing a value or accepting the default results in a separate screen. That screen can be used to view a variety of files or piped command output while the normal top iterative display is paused.

Note: This interactive command is only fully realized when supporting entries have been manually added to the end of the top configuration file. For details on creating those entries, see topic 6b. ADDING INSPECT Entries.

Most of the keys used to navigate the Inspect feature are reflected in its header prologue. There are, however, additional keys available once you have selected a particular file or command. They are familiar to anyone who has used the pager 'less' and are summarized here for future reference.

key	*function*
=	alternate status−line, file or pipeline
/	find, equivalent to 'L' locate
n	find next, equivalent to '&' locate next
<Space>	scroll down, equivalent to <PgDn>
b	scroll up, equivalent to <PgUp>
g	first line, equivalent to <Home>
G	last line, equivalent to <End>

Z :*Change-Color-Mapping*
This key will take you to a separate screen where you can change the colors for the 'current' window, or for all windows. For details regarding this interactive command see topic 4d. COLOR Mapping.

* The commands shown with an asterisk ('*') are not available in Secure mode, nor will they be shown on the level-1 help screen.

4b. SUMMARY AREA Commands

The summary area interactive commands are **always available** in both full−screen mode and alternate−display mode. They affect the beginning lines of your display and will determine the position of messages and prompts.

These commands always impact just the 'current' window/field group. See topic 5. ALTERNATE−DISPLAY Provisions and the 'g' interactive command for insight into 'current' windows and field groups.

C :*Show-scroll-coordinates* toggle
Toggle an informational message which is displayed whenever the message line is not otherwise being used. For additional information see topic 5c. SCROLLING a Window.

l :*Load-Average/Uptime* toggle
This is also the line containing the program name (possibly an alias) when operating in full−screen mode or the 'current' window name when operating in alternate−display mode.

t :*Task/Cpu-States* toggle
This command affects from 2 to many summary area lines, depending on the state of the '1', '2' or '3' command toggles and whether or not top is running under true SMP.

This portion of the summary area is also influenced by the 'H' interactive command toggle, as reflected in the total label which shows either Tasks or Threads.

This command serves as a 4-way toggle, cycling through these modes:
1. detailed percentages by category
2. abbreviated user/system and total % + bar graph
3. abbreviated user/system and total % + block graph
4. turn off task and cpu states display

When operating in either of the graphic modes, the display becomes much more meaningful when

individual CPUs or NUMA nodes are also displayed. See the the ‘1’, ‘2’ and ‘3’ commands below for additional information.

m :*Memory/Swap-Usage* toggle
This command affects the two summary area lines dealing with physical and virtual memory.

This command serves as a 4-way toggle, cycling through these modes:
1. detailed percentages by memory type
2. abbreviated % used/total available + bar graph
3. abbreviated % used/total available + block graph
4. turn off memory display

1 :*Single/Separate-Cpu-States* toggle
This command affects how the ‘t’ command’s Cpu States portion is shown. Although this toggle exists primarily to serve massively-parallel SMP machines, it is not restricted to solely SMP environments.

When you see ‘%Cpu(s):’ in the summary area, the ‘1’ toggle is *On* and all cpu information is gathered in a single line. Otherwise, each cpu is displayed separately as: ‘%Cpu0, %Cpu1, ...’ up to available screen height.

2 :*NUMA-Nodes/Cpu-Summary* toggle
This command toggles between the ‘1’ command cpu summary display (only) or a summary display plus the cpu usage statistics for each NUMA Node. It is only available if a system has the requisite NUMA support.

3 :*Expand-NUMA-Node*
You will be invited to enter a number representing a NUMA Node. Thereafter, a node summary plus the statistics for each cpu in that node will be shown until the ‘1’, ‘2’ or ‘4’ command toggle is pressed. This interactive command is only available if a system has the requisite NUMA support.

4 :*Display-Cpus-Two-Abreast*
This command turns the ‘1’ toggle *Off* for individual cpu display but prints the results two abreast. It requires a terminal with a minimum width of 80 columns. If a terminal’s width is decreased below the minimum while top is running, top reverts to the normal ‘1’ toggle *Off* state.

To avoid truncation when displaying detailed cpu statistcs, as opposed to the graphic representations, a minimum width of 165 columns would be required.

! :*Combine-Cpus-Mode*
This command toggle is intended for massively parallel SMP environments where, even with the ‘4’ command toggle, not all processors can be displayed. With each press of ‘!’ the number of additional cpu’s combined is doubled thus reducing the total number of cpu lines displayed.

For example, with the first press of ‘!’ one additional cpu will be combined and displayed as ‘0-1, 2-3, ...’ instead of the normal ‘%Cpu0, %Cpu1, %Cpu2, %Cpu3, ...’. With a second ‘!’ command toggle two additional cpus are combined and shown as ‘0-2, 3-5, ...’. Then the third ’!’ press, combining four additional cpus, shows as ‘0-4, 5-9, ...’, etc.

Such progression continues until individual cpus are again displayed and impacts both the ‘1’ and ‘4’ toggles (one or two columns). Use the ‘=’ command to exit **Combine Cpus** mode.

Note: If the entire summary area has been toggled *Off* for any window, you would be left with just the **message line**. In that way, you will have maximized available task rows but (temporarily) sacrificed the program name in full−screen mode or the ‘current’ window name when in alternate−display mode.

4c. TASK AREA Commands

The task area interactive commands are **always** available in full−screen mode.

The task area interactive commands are **never available** in alternate−display mode *if* the ‘current’ window’s task display has been toggled *Off* (see topic 5. ALTERNATE−DISPLAY Provisions).

APPEARANCE of task window

J :*Justify-Numeric-Columns* toggle
Alternates between right-justified (the default) and left-justified numeric data. If the numeric data completely fills the available column, this command toggle may impact the column header only.

j :*Justify-Character-Columns* toggle
Alternates between left-justified (the default) and right-justified character data. If the character data completely fills the available column, this command toggle may impact the column header only.

The following commands will also be influenced by the state of the global ‘B’ (bold enable) toggle.

b :*Bold/Reverse* toggle
This command will impact how the ‘x’ and ‘y’ toggles are displayed. It may also impact the summary area when a bar graph has been selected for cpu states or memory usage via the ‘t’ or ‘m’ toggles.

x :*Column-Highlight* toggle
Changes highlighting for the current sort field. If you forget which field is being sorted this command can serve as a quick visual reminder, providing the sort field is being displayed. The sort field might *not* be visible because:
1) there is insufficient *Screen Width*
2) the ‘f’ interactive command turned it *Off*

Note: Whenever Searching and/or Other Filtering is active in a window, column highlighting is temporarily disabled. See the notes at the end of topics 5d. SEARCHING and 5e. FILTERING for an explanation why.

y :*Row-Highlight* toggle
Changes highlighting for "running" tasks. For additional insight into this task state, see topic 3a. DESCRIPTIONS of Fields, the ‘S’ field (Process Status).

Use of this provision provides important insight into your system’s health. The only costs will be a few additional tty escape sequences.

z :*Color/Monochrome* toggle
Switches the ‘current’ window between your last used color scheme and the older form of black-on-white or white-on-black. This command will alter **both** the summary area and task area but does not affect the state of the ‘x’, ‘y’ or ‘b’ toggles.

CONTENT of task window

c :*Command-Line/Program-Name* toggle
This command will be honored whether or not the COMMAND column is currently visible. Later, should that field come into view, the change you applied will be seen.

f | **F** :*Fields-Management*
These keys display a separate screen where you can change which fields are displayed, their order and also designate the sort field. For additional information on these interactive commands see topic 3b. MANAGING Fields.

o | **O** :*Other-Filtering*
You will be prompted for the selection criteria which then determines which tasks will be shown in the 'current' window. Your criteria can be made case sensitive or case can be ignored. And you determine if top should include or exclude matching tasks.

See topic 5e. FILTERING in a window for details on these and additional related interactive commands.

S :*Cumulative-Time-Mode* toggle
When Cumulative mode is *On*, each process is listed with the cpu time that it and its dead children have used.

When *Off*, programs that fork into many separate tasks will appear less demanding. For programs like 'init' or a shell this is appropriate but for others, like compilers, perhaps not. Experiment with two task windows sharing the same sort field but with different 'S' states and see which representation you prefer.

After issuing this command, you'll be informed of the new state of this toggle. If you wish to know in advance whether or not Cumulative mode is in effect, simply ask for help and view the window summary on the second line.

u | **U** :*Show-Specific-User-Only*
You will be prompted for the **uid** or **name** of the user to display. The −u option matches on **effective** user whereas the −U option matches on **any** user (real, effective, saved, or filesystem).

Thereafter, in that task window only matching users will be shown, or possibly no processes will be shown. Prepending an exclamation point ('!') to the user id or name instructs top to display only processes with users not matching the one provided.

Different task windows can be used to filter different users. Later, if you wish to monitor all users again in the 'current' window, re-issue this command but just press <Enter> at the prompt.

V :*Forest-View-Mode* toggle
In this mode, processes are reordered according to their parents and the layout of the COMMAND column resembles that of a tree. In forest view mode it is still possible to toggle between program name and command line (see the 'c' interactive command) or between processes and threads (see the 'H' interactive command).

Note: Typing any key affecting the sort order will exit forest view mode in the 'current' window. See topic 4c. TASK AREA Commands, SORTING for information on those keys.

v :*Hide/Show-Children* toggle
When in forest view mode, this key serves as a toggle to collapse or expand the children of a parent.

The toggle is applied against the first (topmost) process in the 'current' window. See topic 5c. SCROLLING a Window for additional information regarding vertical scrolling.

If the target process has not forked any children, this key has no effect. It also has no effect when not in forest view mode.

SIZE of task window

i :*Idle-Process* toggle
Displays all tasks or just active tasks. When this toggle is *Off*, tasks that have not used any CPU since the last update will not be displayed. However, due to the granularity of the %CPU and TIME+ fields, some processes may still be displayed that *appear* to have used *no* CPU.

If this command is applied to the last task display when in alternate−display mode, then it will not affect the window's size, as all prior task displays will have already been painted.

n | **#** :*Set-Maximum-Tasks*
You will be prompted to enter the number of tasks to display. The lessor of your number and available screen rows will be used.

When used in alternate−display mode, this is the command that gives you precise control over the size of each currently visible task display, except for the very last. It will not affect the last window's size, as all prior task displays will have already been painted.

Note: If you wish to increase the size of the last visible task display when in alternate−display mode, simply decrease the size of the task display(s) above it.

SORTING of task window

For compatibility, this top supports most of the former top sort keys. Since this is primarily a service to former top users, these commands do not appear on any help screen.

command	*sorted-field*	*supported*
A	start time (non-display)	**No**
M	%MEM	Yes
N	PID	Yes
P	%CPU	Yes
T	TIME+	Yes

Before using any of the following sort provisions, top suggests that you temporarily turn on column highlighting using the 'x' interactive command. That will help ensure that the actual sort environment matches your intent.

The following interactive commands will **only** be honored when the current sort field is **visible**. The sort field might *not* be visible because:
1) there is insufficient *Screen Width*
2) the 'f' interactive command turned it *Off*

< :*Move-Sort-Field-Left*
Moves the sort column to the left unless the current sort field is the first field being displayed.

> :*Move-Sort-Field-Right*
Moves the sort column to the right unless the current sort field is the last field being displayed.

The following interactive commands will **always** be honored whether or not the current sort field is visible.

f | **F** :*Fields-Management*
These keys display a separate screen where you can change which field is used as the sort column, among other functions. This can be a convenient way to simply verify the current sort field, when running top with column highlighting turned *Off*.

R :*Reverse/Normal-Sort-Field* toggle
Using this interactive command you can alternate between high-to-low and low-to-high sorts.

Note: Field sorting uses internal values, not those in column display. Thus, the TTY and WCHAN fields will violate strict ASCII collating sequence.

4d. COLOR Mapping

When you issue the 'Z' interactive command, you will be presented with a separate screen. That screen can be used to change the colors in just the 'current' window or in all four windows before returning to the top display.

The following interactive commands are available.
4 upper case letters to select a **target**
8 numbers to select a **color**
normal toggles available
B :bold disable/enable
b :running tasks "bold"/reverse
z :color/mono
other commands available
a/w :apply, then go to next/prior
<Enter> :apply and exit
q :abandon current changes and exit

If you use 'a' or 'w' to cycle the targeted window, you will have applied the color scheme that was displayed when you left that window. You can, of course, easily return to any window and reapply different colors or turn colors *Off* completely with the 'z' toggle.

The Color Mapping screen can also be used to change the 'current' window/field group in either full−screen mode or alternate−display mode. Whatever was targeted when 'q' or <Enter> was pressed will be made current as you return to the top display.

5. ALTERNATE−DISPLAY Provisions

5a. WINDOWS Overview

Field Groups/Windows:
In full−screen mode there is a single window represented by the entire screen. That single window can still be changed to display 1 of 4 different **field groups** (see the 'g' interactive command, repeated below). Each of the 4 field groups has a unique separately configurable**summary ar ea** and its own

configurable **task area**.

In alternate−display mode, those 4 underlying field groups can now be made visible simultaneously, or can be turned *Off* individually at your command.

The summary area will always exist, even if it's only the message line. At any given time only *one* summary area can be displayed. However, depending on your commands, there could be from *zero* to *four* separate task displays currently showing on the screen.

Current Window:
The 'current' window is the window associated with the summary area and the window to which task related commands are always directed. Since in alternate−display mode you can toggle the task display *Off*, some commands might be restricted for the 'current' window.

A further complication arises when you have toggled the first summary area line *Off*. With the loss of the window name (the 'l' toggled line), you'll not easily know what window is the 'current' window.

5b. COMMANDS for Windows

- | _ :*Show/Hide-Window(s)* toggles
The '−' key turns the 'current' window's task display *On* and *Off*. When*On*, that task area will show a minimum of the columns header you've established with the 'f' interactive command. It will also reflect any other task area options/toggles you've applied yielding zero or more tasks.

The '_' key does the same for all task displays. In other words, it switches between the currently visible task display(s) and any task display(s) you had toggled *Off*. If all 4 task displays are currently visible, this interactive command will leave the summary area as the only display element.

*** = | +** :*Equalize/Reset-Window(s)*
The '=' key forces the 'current' window's task display to be visible. It also reverses any active 'i' (idle tasks), 'n' (max tasks), 'u/U' (user filter), 'o/O' (other filter), 'v' (hide children), 'L' (locate) and '!' (combine cpus) commands. Also, if the window had been scrolled, it will be reset with this command. See topic 5c. SCROLLING a Window for additional information regarding vertical and horizontal scrolling.

The '+' key does the same for all windows. The four task displays will reappear, evenly balanced, while retaining any customizations previously applied beyond those noted for the '=' command toggle.

*** A** :*Alternate-Display-Mode* toggle
This command will switch between full−screen mode and alternate−display mode.

The first time you issue this command, all four task displays will be shown. Thereafter when you switch modes, you will see only the task display(s) you've chosen to make visible.

*** a | w** :*Next-Window-Forward/Backward*
This will change the 'current' window, which in turn changes the window to which commands are directed. These keys act in a circular fashion so you can reach any desired window using either key.

Assuming the window name is visible (you have not toggled 'l' *Off*), whenever the 'current' window name loses its emphasis/color, that's a reminder the task display is *Off* and many

commands will be restricted.

* **g** :*Choose-Another-Window/Field-Group*
You will be prompted to enter a number between 1 and 4 designating the field group which should be made the ‘current’ window.

In full−screen mode, this command is necessary to alter the ‘current’ window. In alternate−display mode, it is simply a less convenient alternative to the ‘a’ and ‘w’ commands.

G :*Change-Window/Field-Group-Name*
You will be prompted for a new name to be applied to the ‘current’ window. It does not require that the window name be visible (the ‘l’ toggle to be *On*).

* The interactive commands shown with an asterisk (‘*’) have use beyond alternate−display mode.
=, A, g are always available
a, w act the same with color mapping
and fields management

5c. SCROLLING a Window
Typically a task window is a partial view into a systems’s total tasks/threads which shows only some of the available fields/columns. With these scrolling keys, you can move that view vertically or horizontally to reveal any desired task or column.

Up,**PgUp** :*Scroll-Tasks*
Move the view up toward the first task row, until the first task is displayed at the top of the ‘current’ window. The*Up* arro w key moves a single line while *PgUp* scrolls the entire window.

Down,**PgDn** :*Scroll-Tasks*
Move the view down toward the last task row, until the last task is the only task displayed at the top of the ‘current’ window. The*Down* arro w key moves a single line while *PgDn* scrolls the entire window.

Left,**Right** :*Scroll-Columns*
Move the view of displayable fields horizontally one column at a time.

Note: As a reminder, some fields/columns are not fixed-width but allocated all remaining screen width when visible. When scrolling right or left, that feature may produce some unexpected results initially.

Additionally, there are special provisions for any variable width field when positioned as the last displayed field. Once that field is reached via the right arrow key, and is thus the only column shown, you can continue scrolling horizontally within such a field. See the ‘C’ interactive command below for additional information.

Home :*Jump-to-Home-Position*
Reposition the display to the un-scrolled coordinates.

End :*Jump-to-End-Position*
Reposition the display so that the rightmost column reflects the last displayable field and the bottom task row represents the last task.

Note: From this position it is still possible to scroll *down* and *right* using the arrow keys. This is true until a single column and a single task is left as the only display element.

C :*Show-scroll-coordinates* toggle
Toggle an informational message which is displayed whenever the message line is not otherwise being used. That message will take one of two forms depending on whether or not a variable width column has also been scrolled.

scroll coordinates: y = n/n (tasks), x = n/n (fields)
scroll coordinates: y = n/n (tasks), x = n/n (fields) **+ nn**

The coordinates shown as **n**/**n** are relative to the upper left corner of the 'current' window. The additional '**+ nn**' represents the displacement into a variable width column when it has been scrolled horizontally. Such displacement occurs in normal 8 character tab stop amounts via the right and left arrow keys.

y = n/n (tasks)
The first **n** represents the topmost visible task and is controlled by scrolling keys. The second**n** is updated automatically to reflect total tasks.

x = n/n (fields)
The first **n** represents the leftmost displayed column and is controlled by scrolling keys. The second **n** is the total number of displayable fields and is established with the '**f**' interactive command.

The above interactive commands are **always** available in full−screen mode but **never** available in alternate−display mode if the 'current' window's task display has been toggled *Off*.

Note: When any form of filtering is active, you can expect some slight aberrations when scrolling since not all tasks will be visible. This is particularly apparent when using the Up/Down arrow keys.

5d. SEARCHING in a Window

You can use these interactive commands to locate a task row containing a particular value.

L :*Locate-a-string*
You will be prompted for the case-sensitive string to locate starting from the current window coordinates. There are no restrictions on search string content.

Searches are not limited to values from a single field or column. All of the values displayed in a task row are allowed in a search string. You may include spaces, numbers, symbols and even forest view artwork.

Keying <Enter> with no input will effectively disable the '&' key until a new search string is entered.

& :*Locate-next*
Assuming a search string has been established, top will attempt to locate the next occurrence.

When a match is found, the current window is repositioned vertically so the task row containing that string is first. The scroll coordinates message can provide confirmation of such vertical repositioning (see the 'C' interactive command). Horizontal scrolling, however, is never altered via searching.

The availability of a matching string will be influenced by the following factors.

a. Which fields are displayable from the total available,
see topic 3b. MANAGING Fields.

b. Scrolling a window vertically and/or horizontally,
see topic 5c. SCROLLING a Window.

c. The state of the command/command-line toggle,
see the ‘c’ interactive command.

d. The stability of the chosen sort column,
for example PID is good but %CPU bad.

If a search fails, restoring the ‘current’ window home (unscrolled) position, scrolling horizontally, displaying command-lines or choosing a more stable sort field could yet produce a successful ‘&’ search.

The above interactive commands are **always** available in full−screen mode but **never** available in alternate−display mode if the ‘current’ window’s task display has been toggled *Off*.

Note: Whenever a Search is active in a window, top will turn column highlighting *Off* to prevent false matches on internal non-display escape sequences. Such highlighting will be restored when a window’s search string is empty. See the ‘x’ interactive command for additional information on sort column highlighting.

5e. FILTERING in a Window

You can use this ‘Other Filter’ feature to establish selection criteria which will then determine which tasks are shown in the ‘current’ window. Such filters can be made presistent if preserved in the rcfile via the ’W’ interactive command.

Establishing a filter requires: 1) a field name; 2) an operator; and 3) a selection value, as a minimum. This is the most complex of top’s user input requirements so, when you make a mistake, command recall will be your friend. Remember the Up/Down arrow keys or their aliases when prompted for input.

Filter Basics

1. field names are case sensitive and spelled as in the header
2. selection values need not comprise the full displayed field
3. a selection is either case insensitive or sensitive to case
4. the default is inclusion, prepending ‘!’ denotes exclusions
5. multiple selection criteria can be applied to a task window
6. inclusion and exclusion criteria can be used simultaneously
7. the 1 equality and 2 relational filters can be freely mixed
8. separate unique filters are maintained for each task window

If a field is not turned on or is not currently in view, then your selection criteria will not affect the display. Later, should a filtered field become visible, the selection criteria will then be applied.

Keyboard Summary

o :*Other-Filter* (lower case)
You will be prompted to establish a filter that **ignores case** when matching.

O :*Other-Filter* (upper case)
You will be prompted to establish a **case sensitive** filter.

^O :*Show-Active-Filters* (Ctrl key + 'o')
This can serve as a reminder of which filters are active in the 'current' window. A summary will be shown on the message line until you press the <Enter> key.

= :*Reset-Filtering* in current window
This clears all of your selection criteria in the 'current' window. It also has additional impact so please see topic 4a. GLOBAL Commands.

+ :*Reset-Filtering* in all windows
This clears the selection criteria in all windows, assuming you are in alternate−display mode. As with the '=' interactive command, it too has additional consequences so you might wish to see topic 5b. COMMANDS for Windows.

Input Requirements

When prompted for selection criteria, the data you provide must take one of two forms. There are 3 required pieces of information, with a 4th as optional. These examples use spaces for clarity but your input generally would not.

```
  #1       #2  #3          ( required )
  Field−Name   ?  include−if−value
! Field−Name   ?  exclude−if−value
#4                        ( optional )
```

Items #1, #3 and #4 should be self−explanatory. Item**#2** represents both a required *delimiter* and the *operator* which must be one of either equality ('=') or relation ('<' or '>').

The '=' equality operator requires only a partial match and that can reduce your 'if−value' input requirements. The '>' or '<' relational operators always employ string comparisons, even with numeric fields. They are designed to work with a field's default *justification* and with homogeneous data. When some field's numeric amounts have been subjected to *scaling* while others have not, that data is no longer homogeneous.

If you establish a relational filter and you **have** changed the default Numeric or Character *justification*, that filter is likely to fail. When a relational filter is applied to a memory field and you**ha ve not** changed the *scaling*, it may produce misleading results. This happens, for example, because '100.0m' (MiB) would appear greater than '1.000g' (GiB) when compared as strings.

If your filtered results appear suspect, simply altering justification or scaling may yet achieve the desired objective. See the 'j', 'J' and 'e' interactive commands for additional information.

Potential Problems

These **GROUP** filters could produce the exact same results or the second one might not display anything at all, just a blank task window.

```
GROUP=root      ( only the same results when )
GROUP=ROOT      ( invoked via lower case 'o' )
```

Either of these **RES** filters might yield inconsistent and/or misleading results, depending on the current memory scaling factor. Or both filters could produce the exact same results.

```
RES>9999        ( only the same results when )
!RES<10000      ( memory scaling is at 'KiB' )
```

This **nMin** filter illustrates a problem unique to scalable fields. This particular field can display a maximum of 4 digits, beyond which values are automatically scaled to KiB or above. So while amounts greater than 9999 exist, they will appear as 2.6m, 197k, etc.

```
nMin>9999       ( always a blank task window )
```

Potential Solutions

These examples illustrate how Other Filtering can be creatively applied to achieve almost any desired result. Single quotes are sometimes shown to delimit the spaces which are part of a filter or to represent a request for status (^O) accurately. But if you used them with if-values in real life, no matches would be found.

Assuming field **nTH** is displayed, the first filter will result in only multi-threaded processes being shown. It also reminds us that a trailing space is part of every displayed field. The second filter achieves the exact same results with less typing.

```
!nTH=‘ 1 ’              ( ’ for clarity only )
nTH>1                   ( same with less i/p )
```

With Forest View mode active and the **COMMAND** column in view, this filter effectively collapses child processes so that just 3 levels are shown.

```
!COMMAND=‘      ‘- ’    ( ’ for clarity only )
```

The final two filters appear as in response to the status request key (^O). In reality, each filter would have required separate input. The **PR** example shows the two concurrent filters necessary to display tasks with priorities of 20 or more, since some might be negative. Then by exploiting trailing spaces, the **nMin** series of filters could achieve the failed ‘9999’ objective discussed above.

```
‘PR>20’ + ‘!PR=-’       ( 2 for right result )
‘!nMin=0 ’ + ‘!nMin=1 ’ + ‘!nMin=2 ’ + ‘!nMin=3 ’ ...
```

Note: Whenever Other Filtering is active in a window, top will turn column highlighting *Off* to prevent false matches on internal non-display escape sequences. Such highlighting will be restored when a window is no longer subject to filtering. See the ‘x’ interactive command for additional information on sort column highlighting.

6. FILES

6a. PERSONAL Configuration File

This file is created or updated via the ’W’ interactive command.

The legacy version is written as ‘$HOME/.your−name−4−top’ + ‘rc’ with a leading period.

A newly created configuration file is written as procps/your−name−4−top’ + ‘rc’ without a leading period. The procps directory will be subordinate to either $XDG_CONFIG_HOME when set as an absolute path or the $HOME/.config directory.

While not intended to be edited manually, here is the general layout:

```
global   # line  1: the program name/alias notation
 "       # line  2: id,altscr,irixps,delay,curwin
per ea   # line  a: winname,fieldscur
window   # line  b: winflags,sortindx,maxtasks,etc
 "       # line  c: summclr,msgsclr,headclr,taskclr
global   # line 15: additional miscellaneous settings
 "       # any remaining lines are devoted to optional
 "       # active ’other filters’ discussed in section 5e above
 "       # plus ’inspect’ entries discussed in section 6b below
```

If a valid absolute path to the rcfile cannot be established, customizations made to a running top will be impossible to preserve.

6b. ADDING INSPECT Entries

To exploit the 'Y' interactive command, you must add entries at the **end** of the top personal configuration file. Such entries simply reflect a file to be read or command/pipeline to be executed whose results will then be displayed in a separate scrollable, searchable window.

If you don't know the location or name of your top rcfile, use the 'W' interactive command to rewrite it and note those details.

Inspect entries can be added with a redirected echo or by editing the configuration file. Redirecting an echo risks overwriting the rcfile should it replace (>) rather than append (>>) to that file. Conversely, when using an editor care must be taken not to corrupt existing lines, some of which will contain unprintable data or unusual characters.

Those Inspect entries beginning with a '#' character are ignored, regardless of content. Otherwise they consist of the following 3 elements, each of which *must* be separated by a tab character (thus 2 '\t' total):

.type: literal 'file' or 'pipe'
.name: selection shown on the Inspect screen
.fmts: string representing a path or command

The two types of Inspect entries are *not* interchangeable. Those designated '**file**' will be accessed using fopen and must reference a single file in the '.fmts' element. Entries specifying '**pipe**' will employ popen, their '.fmts' element could contain many pipelined commands and, none can be interactive.

If the file or pipeline represented in your '.fmts' deals with the specific PID input or accepted when prompted, then the format string must also contain the '**%d**' specifier, as these examples illustrate.

.fmts= /proc/*%d*/numa_maps
.fmts= lsof -P -p*%d*

For '**pipe**' type entries only, you may also wish to redirect stderr to stdout for a more comprehensive result. Thus the format string becomes:

.fmts= pmap -x %d*2>&1*

Here are examples of both types of Inspect entries as they might appear in the rcfile. The first entry will be ignored due to the initial '#' character. For clarity, the pseudo tab depictions (^I) are surrounded by an extra space but the actual tabs would not be.

```
# pipe ^I Sockets ^I lsof -n -P -i 2>&1
pipe ^I Open Files ^I lsof -P -p %d 2>&1
file ^I NUMA Info ^I /proc/%d/numa_maps
pipe ^I Log ^I tail -n100 /var/log/syslog | sort -Mr
```

Except for the commented entry above, these next examples show what could be echoed to achieve similar results, assuming the rcfile name was '.toprc'. However, due to the embedded tab characters, each of these lines should be preceded by '**/bin/echo −e**', not just a simple an 'echo', to enable backslash interpretation regardless of which shell you use.

```
"pipe\tOpen Files\tlsof -P -p %d 2>&1" >> ~/.toprc
"file\tNUMA Info\t/proc/%d/numa_maps" >> ~/.toprc
```

```
"pipe\tLog\ttail -n200 /var/log/syslog | sort -Mr" >> ~/.toprc
```

If any inspect entry you create produces output with unprintable characters they will be displayed in either the ^C notation or hexadecimal <FF> form, depending on their value. This applies to tab characters as well, which will show as '^I'. If you want a truer representation, any embedded tabs should be expanded. The following example takes what could have been a 'file' entry but employs a 'pipe' instead so as to expand the embedded tabs.

```
# next would have contained '\t' ...
# file ^I <your_name> ^I /proc/%d/status
# but this will eliminate embedded '\t' ...
pipe ^I <your_name> ^I cat /proc/%d/status | expand −
```

Note: Some programs might rely on *SIGINT* to end. Therefore, if a '**pipe**' such as the following is established, one must use Ctrl-C to terminate it in order to review the results. This is the single occasion where a '^C' will not also terminate top.

```
pipe ^I Trace ^I /usr/bin/strace -p %d 2>&1
```

Lastly, while '**pipe**' type entries have been discussed in terms of pipelines and commands, there is nothing to prevent you from including *shell scripts* as well. Perhaps e ven newly created scripts designed specifically for the 'Y' interactive command.

For example, as the number of your Inspect entries grows over time, the 'Options:' row will be truncated when screen width is exceeded. That does not affect operation other than to make some selections invisible. However, if some choices are lost to truncation but you want to see more options, there is an easy solution hinted at below.

```
Inspection Pause at pid ...
Use:  left/right then <Enter> ...
Options:  help  1  2  3  4  5  6  7  8  9  10  11 ...
```

The entries in the top rcfile would have a number for the '.name' element and the 'help' entry would identify a shell script you've written explaining what those numbered selections actually mean. In that way, many more choices can be made visible.

6c. SYSTEM Configuration File

This configuration file represents defaults for users who have not saved their own configuration file. The format mirrors exactly the personal configuration file and can also include 'inspect' entries as explained above.

Creating it is a simple process.

1. Configure top appropriately for your installation and preserve that configuration with the 'W' interactive command.

2. Add and test any desired 'inspect' entries.

3. Copy that configuration file to the */etc/* directory as '**topdefaultrc**'.

6d. SYSTEM Restrictions File

The presence of this file will influence which version of the help screen is shown to an ordinary user.

More importantly, it will limit what ordinary users are allowed to do when top is running. They will not be

able to issue the following commands.
k Kill a task
r Renice a task
d or s Change delay/sleep interval

This configuration file is not created by top. Rather, it is created manually and placed it in the */etc/* directory as '**toprc**'.

It should have exactly two lines, as shown in this example:
s # line 1: secure mode switch
5.0 # line 2: delay interval in seconds

7. STUPID TRICKS Sampler

Many of these tricks work best when you give top a scheduling boost. So plan on starting him with a nice value of −10, assuming you've got the authority.

7a. Kernel Magic

For these stupid tricks, top needs full−screen mode.

- The user interface, through prompts and help, intentionally implies that the delay interval is limited to tenths of a second. However, you're free to set any desired delay. If you want to see Linux at his scheduling best, try a delay of .09 seconds or less.

  For this experiment, under x-windows open an xterm and maximize it. Then do the following:
  . provide a scheduling boost and tiny delay via:
  nice -n -10 top -d.09
  . keep sorted column highlighting *Off* so as to
  minimize path length
  . turn *On* reverse row highlighting for emphasis
  . try various sort columns (TIME/MEM work well),
  and normal or reverse sorts to bring the most
  active processes into view

  What you'll see is a very busy Linux doing what he's always done for you, but there was no program available to illustrate this.

- Under an xterm using 'white-on-black' colors, on top's Color Mapping screen set the task color to black and be sure that task highlighting is set to bold, not reverse. Then set the delay interval to around .3 seconds.

  After bringing the most active processes into view, what you'll see are the ghostly images of just the currently running tasks.

- Delete the existing rcfile, or create a new symlink. Start this new version then type 'T' (a secret key, see topic 4c. Task Area Commands, SORTING) followed by 'W' and 'q'. Finally, restart the program with −d0 (zero delay).

  Your display will be refreshed at three times the rate of the former top, a 300% speed advantage. As top climbs the TIME ladder, be as patient as you can while speculating on whether or not top will ever reach the top.

7b. Bouncing Windows

For these stupid tricks, top needs alternate−display mode.

- With 3 or 4 task displays visible, pick any window other than the last and turn idle processes *Off* using the ‘i’ command toggle. Depending on where you applied ‘i’, sometimes several task displays are bouncing and sometimes it’s like an accordion, as top tries his best to allocate space.

- Set each window’s summary lines differently: one with no memory (‘m’); another with no states (‘t’); maybe one with nothing at all, just the message line. Then hold down ‘a’ or ‘w’ and watch a variation on bouncing windows −− hopping windows.

- Display all 4 windows and for each, in turn, set idle processes to *Off* using the ‘i’ command toggle. You’ve just entered the "extreme bounce" zone.

7c. The Big Bird Window

This stupid trick also requires alternate−display mode.

- Display all 4 windows and make sure that 1:Def is the ‘current’ window. Then, keep increasing window size with the ‘n’ interactive command until all the other task displays are "pushed out of the nest".

  When they’ve all been displaced, toggle between all visible/invisible windows using the ‘_’ command toggle. Then ponder this:
  is top fibbing or telling honestly your imposed truth?

7d. The Ol’ Switcheroo

This stupid trick works best without alternate−display mode, since justification is active on a per window basis.

- Start top and make COMMAND the last (rightmost) column displayed. If necessary, use the ‘c’ command toggle to display command lines and ensure that forest view mode is active with the ‘V’ command toggle.

  Then use the up/down arrow keys to position the display so that some truncated command lines are shown (‘+’ in last position). You may have to resize your xterm to produce truncation.

  Lastly, use the ‘j’ command toggle to make the COMMAND column right justified.

  Now use the right arrow key to reach the COMMAND column. Continuing with the right arrow key, watch closely the direction of travel for the command lines being shown.

  some lines travel left, while others travel right

  eventually all lines will Switcheroo, and move right

8. BUGS

Please send bug reports to ⟨procps@freelists.org⟩.

9. SEE Also

free(1), **ps**(1), **uptime**(1), **atop**(1), **slabtop**(1), **vmstat**(8), **w**(1)