

# DBA Hacker's guide to Solaris

Not just for DBAs: this is to remind anyone of syntax and options.

1. [Handy Solaris commands](#) HW config, swap, patches etc.
2. [Real-time performance monitoring](#) vmstat, iostat, mpstat etc.
3. [Unix Commands](#) find, awk, sed, grep etc.
4. [vi editor commands](#) the 22 most useful vi commands
5. [Unix file permissions explained](#) sounds basic, but you may learn something new here!
6. [The Solaris Name Service explained](#) how hostname and username lookups work

## 1. Handy Solaris commands

- show hardware config (memory, CPUs etc) :  
`/usr/platform/sun4u/sbin/prtdiag -v |more`  
`/usr/sbin/psrinfo`
- show kernel settings:  
`/usr/sbin/sysdef |more`  
`more /etc/system`
- show list of Solaris patches:  
`showrev -p |more`
- show software installed:  
`pkginfo |more`
- check swap space:  
`/usr/sbin/swap -s`
- check system load average: an uptime more than double the number of CPUs is getting a bit busy  
`uptime`
- top CPU users:  
`/usr/ucb/ps uax |head`
- top memory users:  
`/usr/ucb/ps vax |head`
- check disk space:  
`df -k` (shows all mounted filesystems -including NFS etc)  
`df -lk` (shows only local filesystems)  
`df -k -Fufs` (shows only local Sun UFS filesystems (normal hard disks))  
`df -k .` (show just the filesystem you are currently in)

## 2. Real-time performance monitoring:

- **memory usage :**  
`vmstat 5` -look at memory/free field and page/sr field. Ignore the first line output as it's historical  

procs	memory	page	disk	faults	cpu
-------	--------	------	------	--------	-----

```

r b w swap free re mf pi po fr de sr s0 s1 s6 s3 in sy cs us sy id
0 0 83 4456 456 1 431 266 70 167 0 35 6 6 0 2 523 567 31 14 9 76
0 0 62 3588464 46824 0 196 64 0 0 0 0 5 4 0 0 606 9743 882 86 7 7
0 0 62 3587960 42672 1 552 41 1 1 0 0 2 2 0 0 789 5488 1040 84 7 9
0 1 62 3584704 38848 0 471 3 38 38 0 0 5 5 0 1 1426 5270 968 64 9 27
0 0 62 3586464 38456 0 451 0 0 0 0 0 2 2 0 0 929 6039 1265 70 6 24

```

Also make sure that `cpu/us` is at least double `cpu/sy`

- **disk busy-ness**

```

mpstat 5 (Or iostat -c 5 )
CPU minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt
idl
0 221 3 544 227 75 582 61 31 28 7 267 18 12 2
68
2 209 2 446 395 178 328 37 31 32 6 299 11 7 2
80

```

-look at the `wt` field, this is wait-for-I/O which can be network or disk I/O, and should not be more than 30-40 (percent)

To see individual disk performance and find slow ('hot') disks :

```

iostat -d 5
sd0          sd1          sd6          sd37
kps tps serv kps tps serv kps tps serv kps tps serv
123 6 44 123 6 42 0 0 42 66 2 8
33 1 3 37 1 1 0 0 0 3 0 5

```

-check the `serv` column for each disk: this is the disk service time in milliseconds.

However `iostat` by default shows only the first four disks: if you have more use `-x` and/or `-l` options :

```

iostat -xdn1 7 5 (e.g. if you have seven hard disks)
extended device statistics
r/s w/s kr/s kw/s wait actv wsvc_t asvc_t %w %b device
0.4 1.6 3.2 70.4 0.0 0.0 0.0 2.7 0 1 c0t0d0
0.2 1.6 1.6 70.4 0.0 0.0 0.0 3.0 0 1 c0t1d0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0 0 c0t6d0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0 0 c1t8d0
88.8 19.8 1276.8 158.4 0.0 0.6 0.0 5.7 0 51 c1t9d0
0.0 0.4 0.0 100.8 0.0 0.0 0.0 17.4 0 1 c1t10d0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0 0 c1t11d0

```

### 3. Unix Commands

- **at** - run a command or script once in the future

```

bash$ at 0815 tomorrow
at> rm $HOME/zxc
at> (type Control-D)
commands will be executed using /usr/bin/bash
job 984032100.a at Thu Mar 8 06:15:00 2001

```

- **awk**

- **cron**

`crontab -l` to see your crontab

`crontab -e` to edit it (NB but first do `EDITOR=vi` (or `EDITOR=emacs`) ; `export EDITOR` )

Remember fields are : minute hour day month day\_of\_week (0=Sun)

0-59 0-23 1-31 1-12 0-6

e.g. run a script only on Tuesdays to Fridays, at 10:12 am, 3:12pm, 4:12pm and 5:12pm  
12 10,15-17 \* \* 2-5/home/oracle/bin/script

- **du -show disk space usage**

`du -k` -show disk usage in kB in all subdirectories below the current one

`du -sk` -show just the total of everything below the current dir

`du -sk *` -show individual totals for each file or directory in the current dir

- **find**

find things you can't remember the exact name of :

`find /usr/lib -name "*socket*"`

find big files (more than 20000 blocks, ie about 10MB )

`find . -size +20000`

find files or directories modified in last day

`find . -mtime -1`

find files not accessed (read) in 45 days

`find /tmp -atime +45`

Add `-ls` to the end of any of the above commands to see a full `ls -l` listing of the files found.

You can combine multiple options , e.g look for which big files have filled up a filesystem recently

`find . -size +20000 -mtime -1 -ls`

Or combine options using an "OR" syntax, e.g. find files which were modified either less than 1 week ago or more than 2 weeks ago

`find . \( -mtime +14 -o -mtime -7 \) -ls`

You can send the output of find to another command : the xargs command is ideal for this:

e.g. interactively delete all core files under your \$HOME

`find ~ -type f -name core |xargs rm -i`

or look for a particular word in all your .c files:

`find . -name "*.c" |xargs grep -l libsocket`

- **grep** and its more flexible and faster cousins **egrep** and **fgrep**

Search for either of two strings in all log files:

`egrep "error|PROBLEM" *log`

case-insensitive search

`fgrep -i error *log` (will find Error, ERROR, etc.)

just see the filenames where the search text was found

`grep -l error *log`

- **sort**

`-n` sorts by number instead of alphabetically

e.g. `cd /export/home; du -sk * | sort -nr` lists who is using most space in their home directory

- **tar**

e.g. to copy a directory tree, preserving symlinks:

`cd fromdir; tar cf - . | (cd todir; tar xpf -)`

- **tr** translate characters

convert lower-case to upper:

`cat myfile | tr [a-z] [A-Z]`

change colons to newlines:

```
tr : "\n" < myfile > newfile
```

- **who** show who is currently logged on and where from
- **w** show what people are doing

other useful Unix commands for you to check out:

```
cut, paste, split
```

```
diff, sdiff, cmp, dircmp
```

```
head, tail, tail -f
```

```
id, groups
```

## 4. vi commands

The basic vi commands:

- |                    |  |          |   |
|--------------------|--|----------|---|
| <b>i</b>           | insert before current character                          | <b>a</b> | append after current char               |
| <b>I</b>           | insert at beginning of line                              | <b>A</b> | append after end of line                |
| <b>x</b>           | delete current character                                 | <b>X</b> | backspace-delete (deletes char to left) |
| <b>dd</b>          | delete current line                                      |          |   |
| <b>22dd</b>        | delete 22 lines  | <b>u</b> | undo the last thing you did             |
| <b>P</b>           | paste those 22 lines below the current line              |          |   |
| <b>12yy</b>        | copy 12 lines into the 'clipboard' for later pasting     |          |   |
| <b>&lt;Esc&gt;</b> | to get out of editing (insert/append) mode               |          |   |
| <b>:wq</b>         | to save ('write') and quit (:q! to quit without writing) |          |   |

### vi moving-around Timesavers:

using h,j,k,l to move left,down,up and right is quicker than using the arrow keys (once you get used to it!).

- |              |                                |              |                          |           |                   |
|--------------|--------------------------------|--------------|--------------------------|-----------|-------------------|
| <b>w</b>     | move one word forward          | <b>b</b>     | move back a word         | <b>e</b>  | go to end of word |
| <b>Ctl-F</b> | move Forward a whole page      | <b>Ctl-B</b> | move Back a page         |           |                   |
| <b>0</b>     | (zero) go to beginning of line |              |                          | <b>\$</b> | go to end of line |
| <b>:242</b>  | go to line 242                 | <b>Ctl-G</b> | see what line you are on |           |                   |

### vi modifying things:

- |           |             |            |                |          |                       |
|-----------|-------------|------------|----------------|----------|-----------------------|
| <b>cw</b> | change word | <b>3cw</b> | change 3 words | <b>C</b> | change to end-of-line |
|-----------|-------------|------------|----------------|----------|-----------------------|

<b>dw</b>	delete word		<b>D</b>	delete to end-of-line
<b>cc</b>	replace current line with what you type next			
<b>r</b>	replace one character	<b>R</b>	endless replace (like over-typing)	
<b>o</b>	open/add a new line below the current one		<b>O</b>	open new line above current one
<b>xp</b>	swap two characters (e.g. when you make a typo)			
<b>/bob</b>	search forward for 'bob'	<b>n</b>	repeat previous search	<b>?</b> search backwards

**Probably the handiest vi command:**

. (dot) -repeat your last command

**useful extra vi commands:**

~ swap case of current character (capitalize or lower-case)  
**ct1-L** re-draw the screen (e.g when something from a background process writes to your screen and messes up your vi window)  
**:set nu** show line numbers (:set nonu to remove line numbers)  
**:set list** show hidden characters, line endings etc. (: set nolist)

Global replace:

**:%s/old/new/g** (% means all lines)  
 or **:g/old/s//new/g**  
 or **:%s/old/new/gc** (c is to confirm each replacement, type 'y' to accept)

**5. Unix file permissions**

You probably already know the basics:

Each user in Unix belongs to at least one group, each file or directory on the system belongs to one user and one group. When you do an `ls -l` on a file you see what permissions the file owner, group owner and everyone else ('world' or 'others') have on it

```
-rwxr-xr-- 1 robins devteam 180 Mar 8 13:50 instbb
```

so on the file `instbb` there is full access (rwx- read,write,execute) for the owner (robins), read and execute for the group devteam and read-only for everyone else.

What you may not know is that whether someone can delete that file is not determined by the file permissions, but by the permissions of the directory the file is in :

```
bash$ ls -al
total 598
drwxrwxr-x 3 robins devteam 512 Mar 8 12:02 .
drwxr-xr-x 24 root wheel 1024 Mar 8 12:02 ..
-rwxr-xr-- 1 robins devteam 180 Mar 8 13:50 instbb
```

In this case the directory (.) is group writeable, which means anyone in the group devteam can delete the file `instbb`. Although they can't modify it, they could copy it to a new file in that directory, modify it, then delete the original and rename the new file as `instbb`. So the file isn't as secure as it may appear..

**6. Solaris Name services**

Solaris provides a mechanism for getting hostnames and usernames etc from several sources (e.g DNS, NIS, or the traditional `/etc/hosts` file) : the file `/etc/nsswitch.conf` , which may contain

```
hosts: files dns
```

this means that when you try to access a remote host by name (e.g. ping neptune) it will look for neptune first in /etc/hosts, then do an lookup in DNS (nslookup using the server specified in /etc/resolv.conf)

Similarly for usernames, you may have

```
passwd: files nis
```

So when you run a command which refers to a username (e.g. cd ~oracle ), it first looks in /etc/passwd then does a lookup in NIS (ypmatch oracle passwd).

However you can look up hosts and users without worrying about where they are stored, using the `getent` command :

```
bash$ getent passwd oracle
oracle:##oracle:3008:5001:Oracle
DBA:/export/home/oracle:/usr/local/bin/bash
bash$ getent hosts www.inkq.com
192.168.245.12 www.inkq.com
bash$ getent hosts 10.0.0.91
10.0.0.91 plop.inkq.com plop mailhost
```