

# Computer Engineering Fun Logics (Engineered in Linux)

iNode (Yes, its not typo.Don't read as iPhone)

**inode**

Surprisingly inode term date backs to history in 19XX

```
INODE(7)                                Linux Programmer's Manual                                INODE(7)

NAME
    inode - file inode information

DESCRIPTION
    Each file has an inode containing metadata about the file.  An applica-
    tion can retrieve this metadata using stat(2) (or related calls), which
    returns a stat structure, or statx(2), which returns a statx structure.
```

# A file which never grows its size on physical disk

**/dev/null**

No matter how much you write into this file, it consumes it all and surprisingly never grows file size and hence disk space never wasted.

Application:

```
jigarpandya@aharnish:~$ ls -l /dev/null  
crw-rw-rw- 1 root root 1, 3 Aug 23 08:40 /dev/null
```

Whenever a certain server/application program is generating output which you just want to ignore, redirect here. But the question is why would you want to ignore output? Funny isn't it?

# Something is both parent and child for itself

## File system root / in Linux directory structure

. refers to current directory while .. refers to parent directory

Root (/) . and .. both are itself. Note the inode number 2.

```
root@aharnish:/# ls -lid . ..  
2 drwxr-xr-x 28 root root 4096 Jul 11 2023 .  
2 drwxr-xr-x 28 root root 4096 Jul 11 2023 ..
```

Applications: **Chicken and Egg situation** This actually made possible to define root itself. Directories can have sub-directories which themselves are directories as such. But recursion base case is interesting here. Also then who is inode number 1? Lol.

Functions: always succeeds, returns twice , never returns

umask

```
RETURN VALUE
    This system call always succeeds
```

fork()

```
RETURN VALUE
    On success, the PID of the child process is returned in the parent, and 0 is returned in the child. On failure, -1 is returned
```

exit()

```
RETURN VALUE
    These functions do not return.
```

# A data structure as well as a file ..

## /proc

Process information pseudo-filesystem

This provides an interface to kernel data structures and mounted at a directory  
/proc

```
jigarpandya@aharnish:~$ ls -ld /proc/meminfo  
-r--r--r-- 1 root root 0 Aug 23 08:41 /proc/meminfo
```

Application: Read data structures of a program (in this case OS Kernel) as you are reading from a file

# man man : Things which teach also need to be learnt!

man command stands for user manual in Linux of commands/apis and more  
Interestingly man man shows manual of itself too.

```
MAN(1)                                Manual pager utils                                MAN(1)

NAME
    man - an interface to the on-line reference manuals

SYNOPSIS
```

Application: If something helps to learn about other things people still need to learn that. How about teacher teaches about itself too!

# 0 is FALSE and 0 is SUCCESS too

## EXIT\_SUCCESS

The C standard specifies two constants, `EXIT_SUCCESS` and `EXIT_FAILURE`, that may be passed to `exit()` to indicate successful or unsuccessful termination, respectively.

The use of `EXIT_SUCCESS` and `EXIT_FAILURE` is slightly more portable (to non-UNIX environments) than the use of 0 and some nonzero value like 1 or -1. In particular, VMS uses a different convention.

echo \$? No no no ... don't think success is false.

Success is one, failures can be many.

```
#include <stdio.h>
int main()
{
    //Try to get this right
    if(0)
        printf("Me True\n");
    else
        printf("You False\n");

    if(0)
        printf("You False");
    else
        printf("Me True");
}
```

```
jigarpandya@aharnish:~/my-dev$ ./a.out
You False
Me Truejigarpandya@aharnish:~/my-dev$
```



# Zombies exist and God too ... Even in computer world!

man ps

```
Processes marked <defunct> are dead processes (so-called "zombies")  
that remain because their parent has not destroyed them properly.  
These processes will be destroyed by init(8) if the parent process  
exits.
```

Init

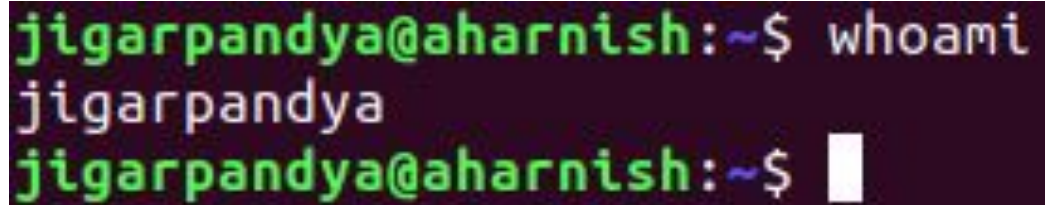
Then who manages init?

```
systemd is a system and service manager for Linux operating systems.  
When run as first process on boot (as PID 1), it acts as init system  
that brings up and maintains userspace services.
```

To support/Support to those who forget who they are :

who

whoami

A terminal window with a dark background and green text. The prompt 'jigarpandya@aharnish:~\$' is followed by the command 'whoami'. The output is 'jigarpandya' on the next line. The prompt is repeated on the third line, followed by a white cursor block.

```
jigarpandya@aharnish:~$ whoami
jigarpandya
jigarpandya@aharnish:~$ █
```

Application:

When working on multiple computers at once and performing many tasks via different roles it comes handy to remind yourself who you are and what you should be doing and what not. Ha ha ...

# Everything is of importance and wants to identify by same

Administrator/Super User is known as root

Root is / beginning of file system

Root is also name of a user

```
jigarpandya@aharnish:~$ cat /etc/passwd | grep root  
root:x:0:0:root:/root:/bin/bash  
jigarpandya@aharnish:~$
```

Root is also a directory

```
jigarpandya@aharnish:~$ ls -ld /root  
drwx----- 36 root root 4096 Aug 23 09:14 /root
```

Am I missing anything ???

# Immunization is everywhere

**nohup**

No hang up

NOHUP(1)	User Commands	NOHUP(1)
<b>NAME</b>		
nohup - run a command immune to hangups, with output to a non-tty		

# One man army is possible

id (uid,gid) ,adduser

```
NAME
      adduser, addgroup - add a user or group to the system
```

```
jigarpandya@aharnish:~$ id
uid=1000(jigarpandya) gid=1000(jigarpandya)
```

```
jigarpandya@aharnish:~$ cat /etc/passwd | grep jigarpandya
jigarpandya:x:1000:1000:JigarPandya,,,:/home/jigarpandya:/bin/bash
jigarpandya@aharnish:~$
```

```
jigarpandya@aharnish:~$ cat /etc/group | grep jigarpandya | grep 1000
jigarpandya:x:1000:
jigarpandya@aharnish:~$
```

# Thanks

If you have made through HERE

You have learnt a few of the Linux and Computer Engineering Concepts actually.

Happy Learning !!!

By the way there is something in Linux called HERE DOCUMENT. May wanna learn... have fun.