1

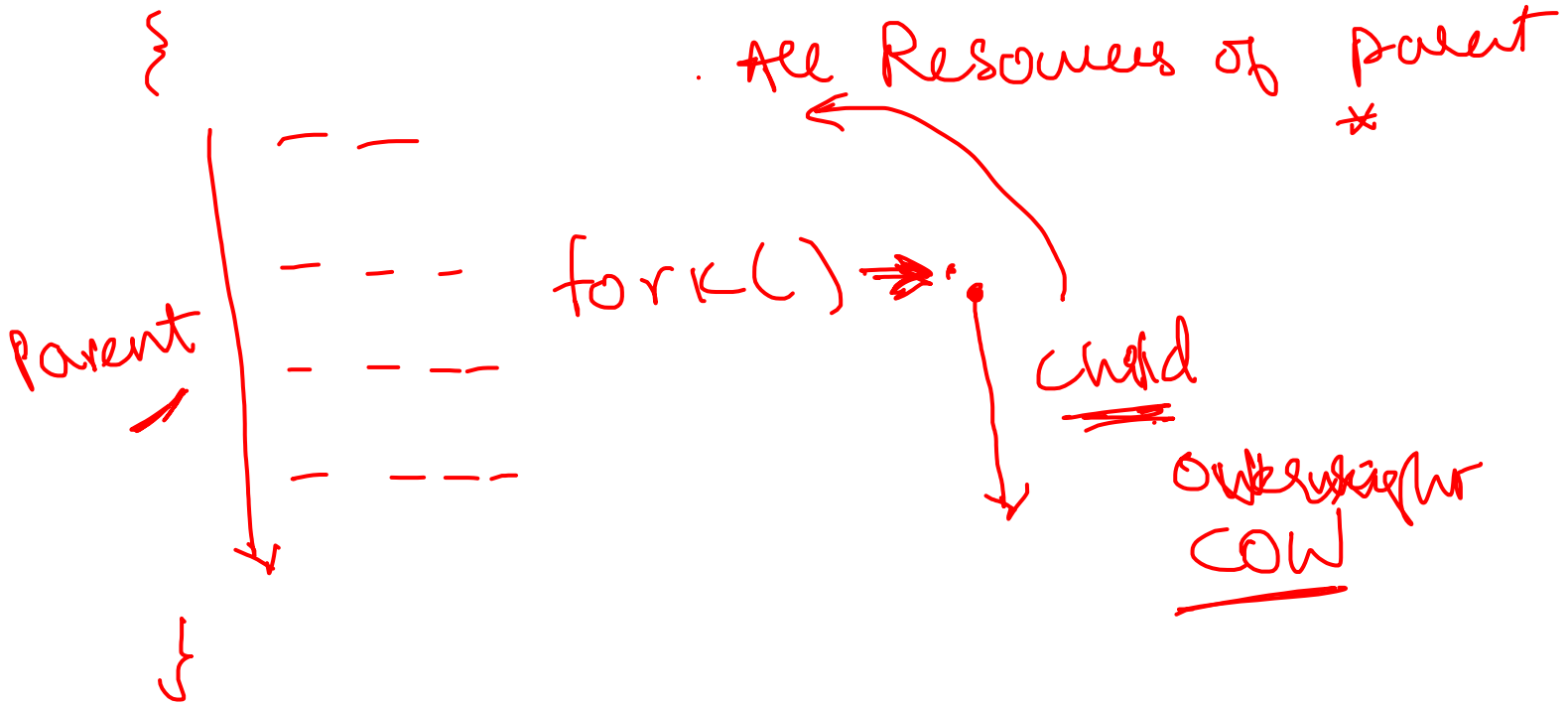
main.c → Program

./main ↔ process

→ Parent process

Process mgt

↳ fork()



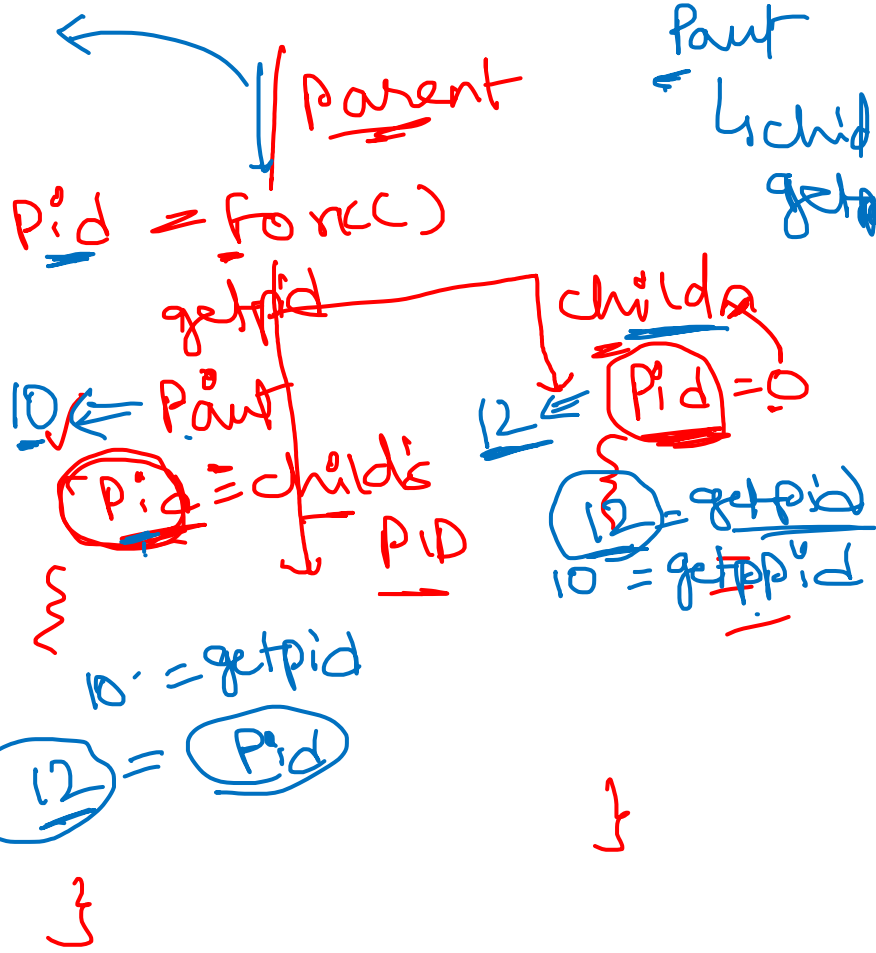
1.

{

Parent : Hello world "

fork

} printf(" Hello world "), Pid = fork()



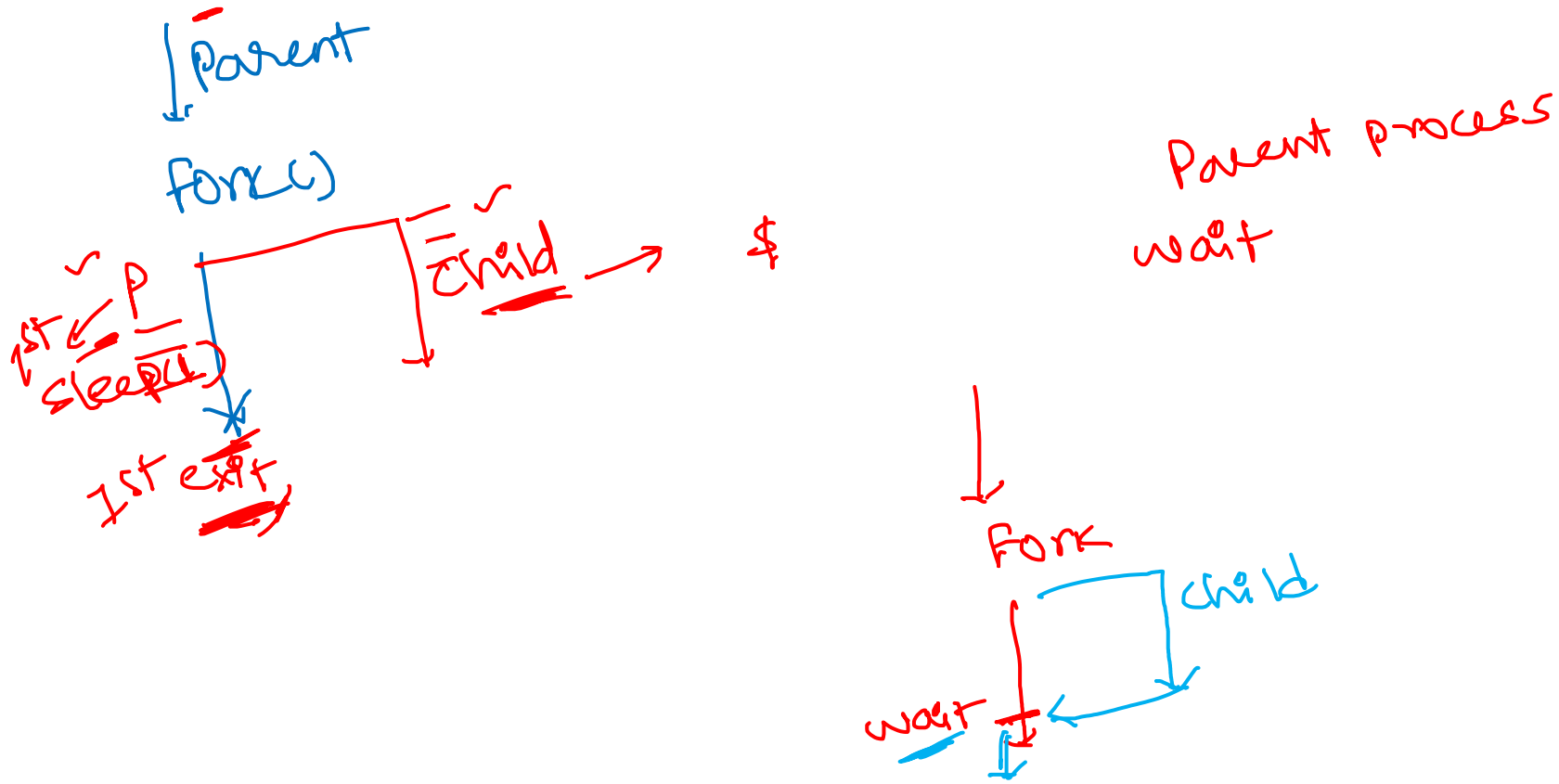
example1

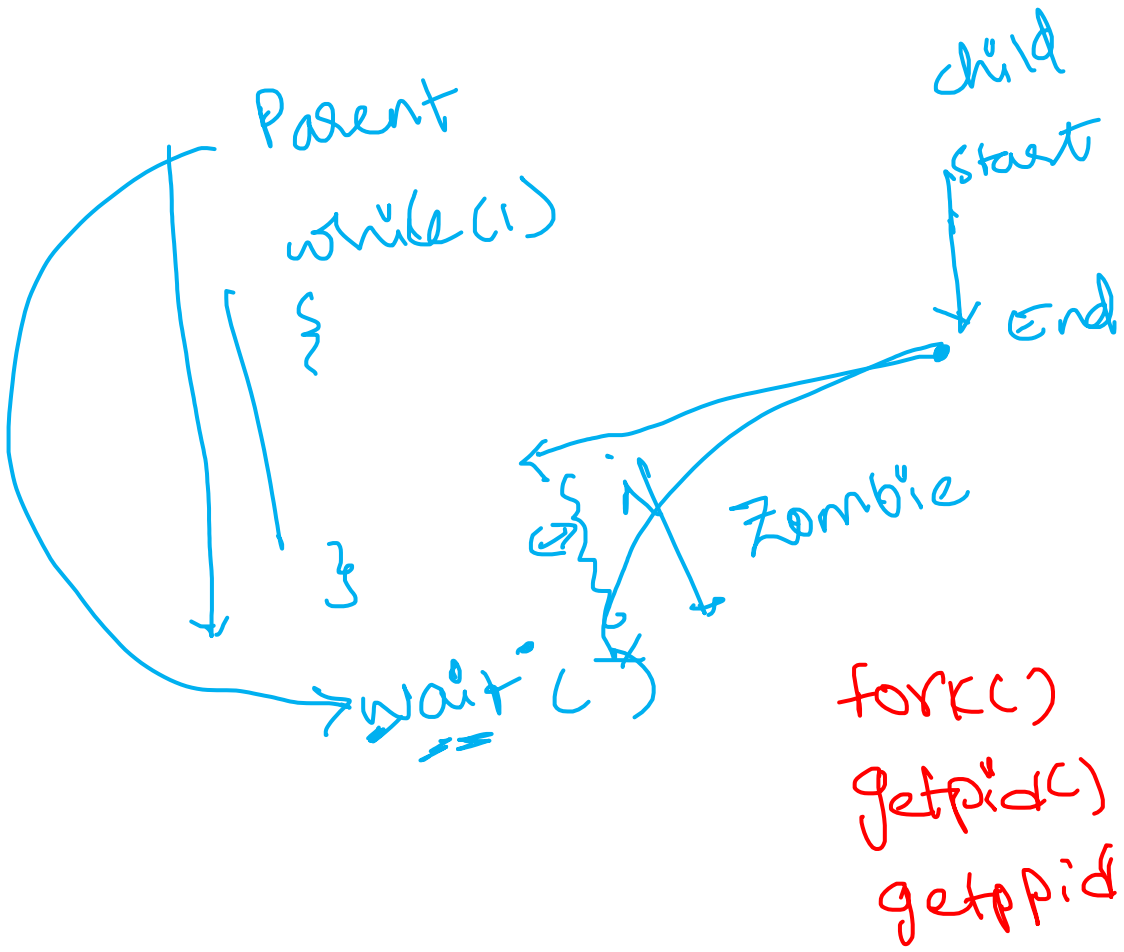
./example1

Pid

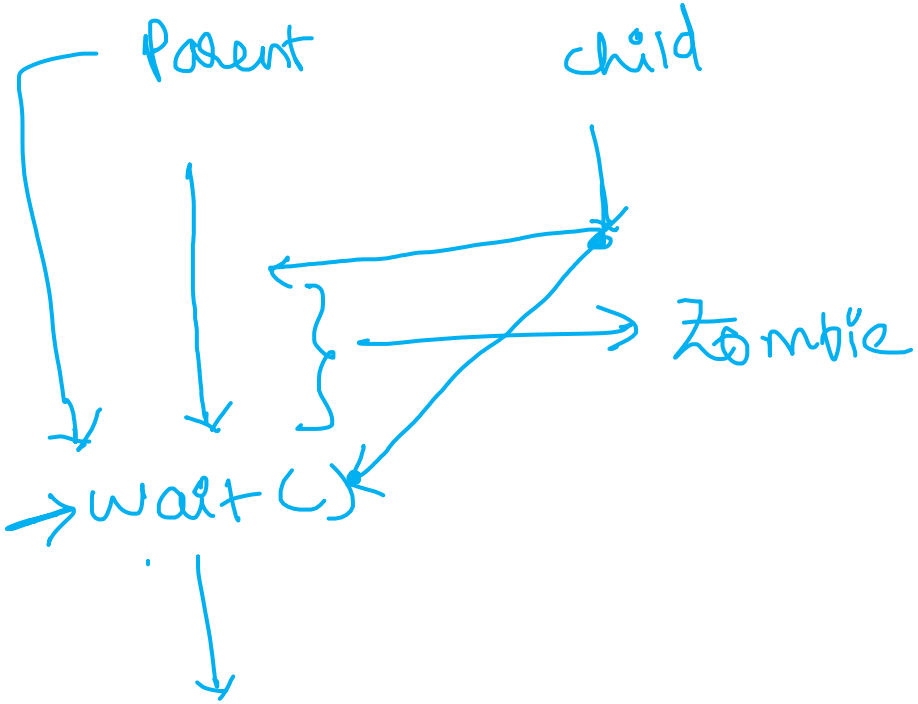
./example1

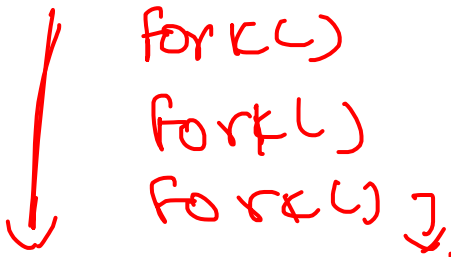
Pid

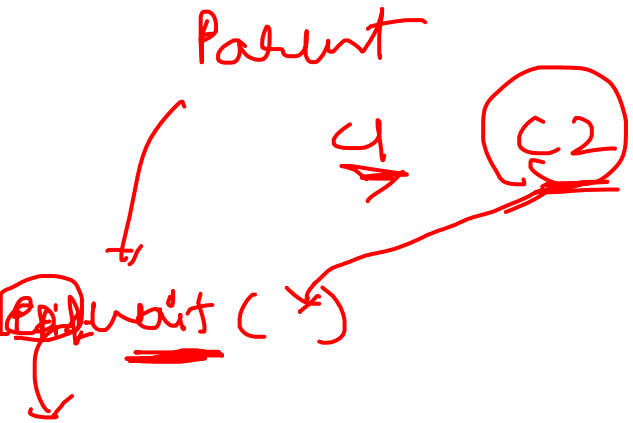

Parent
fork()
child
P
1st
sleep()
1st exit
$
Parent process
wait
fork
child
wait

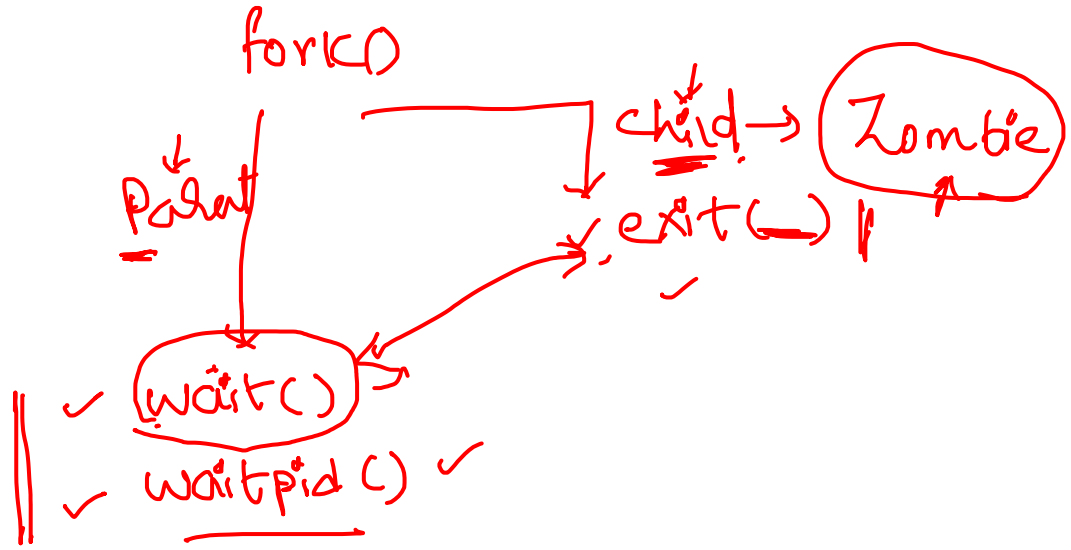


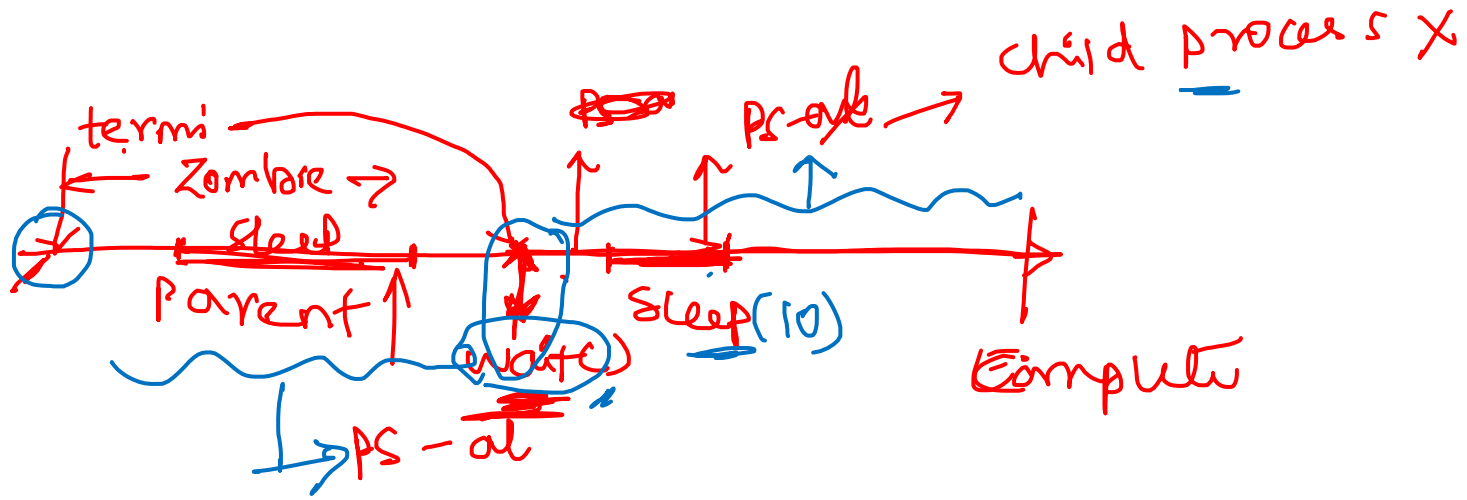
fork()
getpid()
getppid()




fork()
fork()
fork()
printf("Hello world");


Parent
C2
wait()


fork()
Parent
child
Zombie
exit()
wait()
waitpid()


child process X
termi
Zombie
sleep
parent
wait()
ps -al
sleep(10)
Computer

32 bit / 64 bit arch

Databus
32 bits

Databus
64 bit

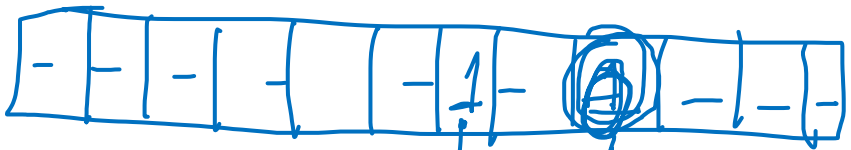
Registers
32 bits

8085 ⇒ 8 bit arch
A B - - -
8 bits

86 ⇒ 16 bit
AX

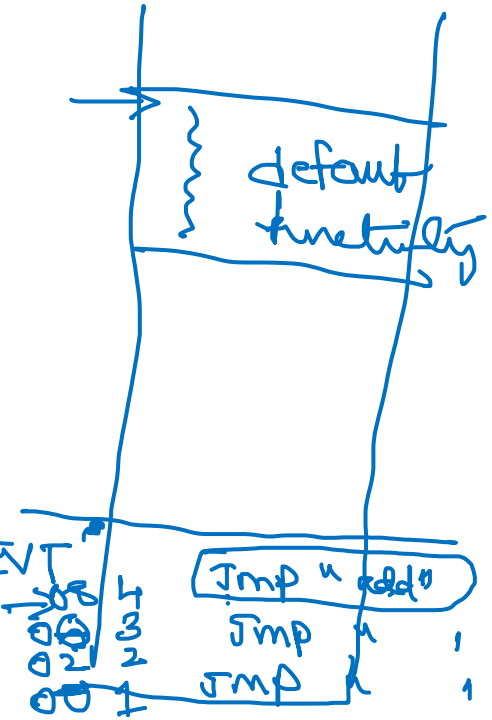
80386
32 bit EAX

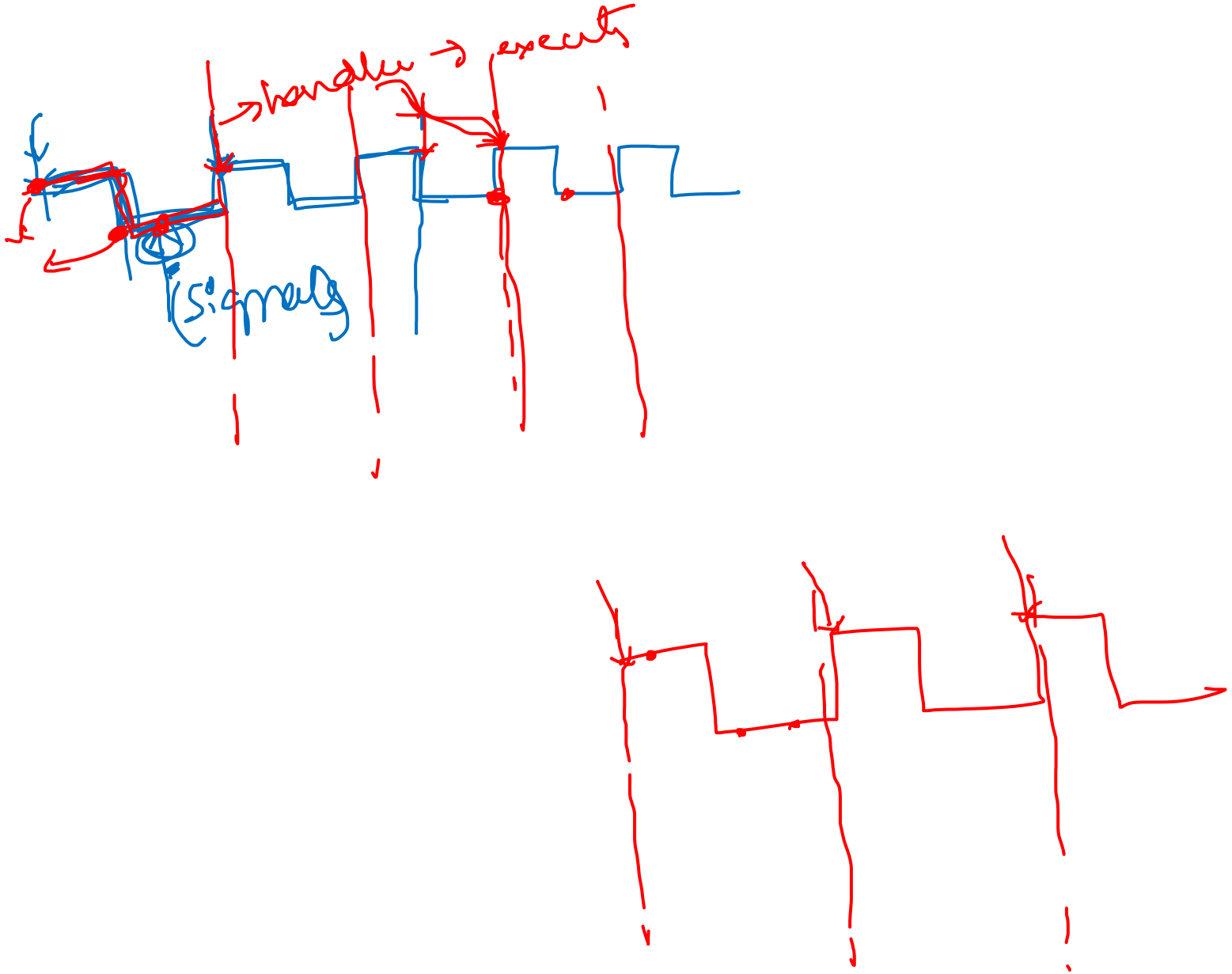
Interrupts
64 Interrupts

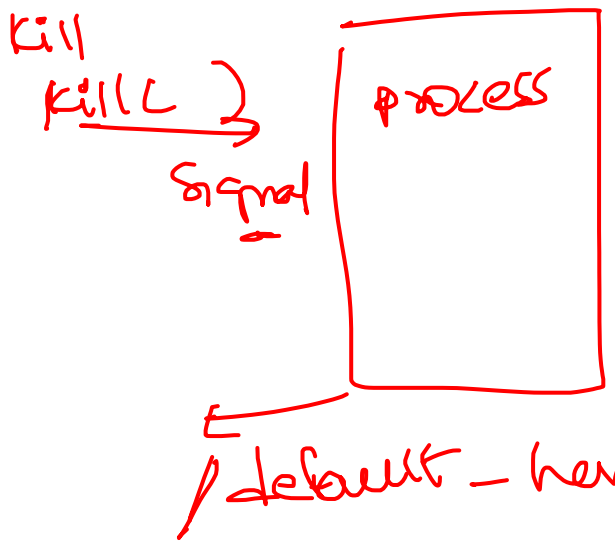


4 x 2 = 8

default functionality → executed




handle
execute
signal

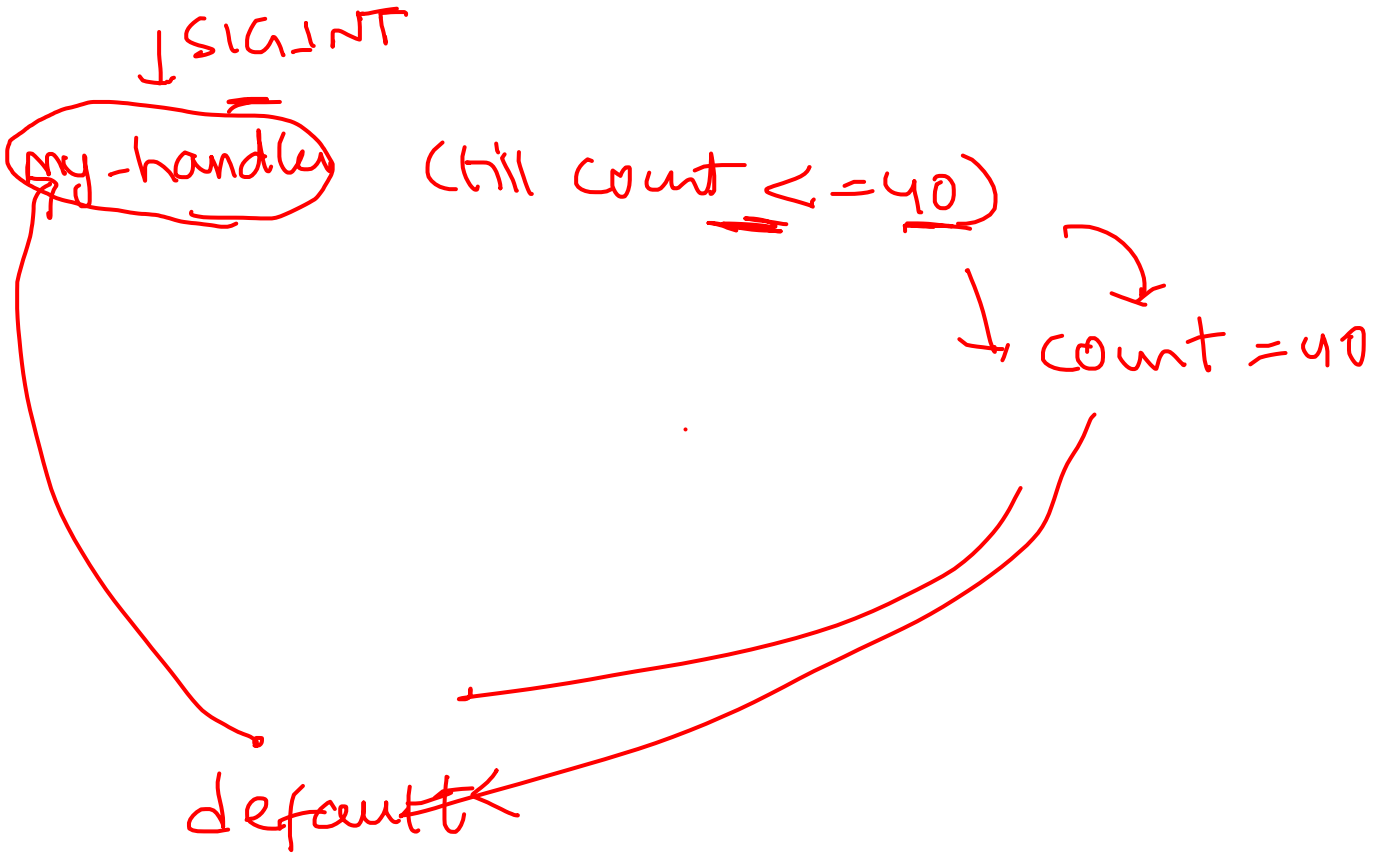


↳ change this default behaviour

signal() ——— step 1: (Register for which you are going to behaviour

- SIG-IGN
- SIG-DFL
- handler

——— step 2: : How to change it


SIGINT
my-handler
(till count <=40)
count =40
default

Exec

→ signal(SIGSTOP, SIG_IGN) ✗

```
while(1)
{
    p ("Hw");
}
```

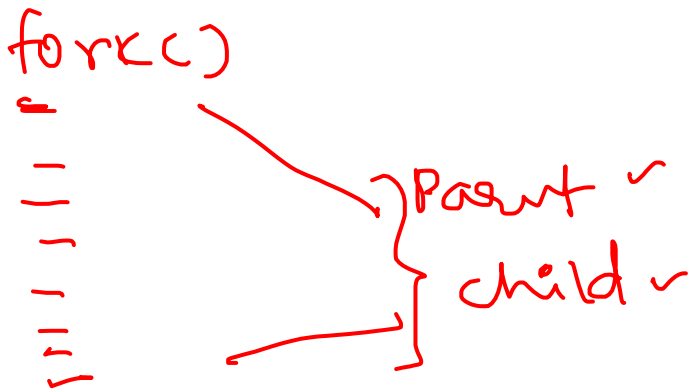
fork(), wait(), exit()

system - - - - kill, kill(), signal()

kill, kill() sends the signal

signal() change the default behaviour of a signal

exec

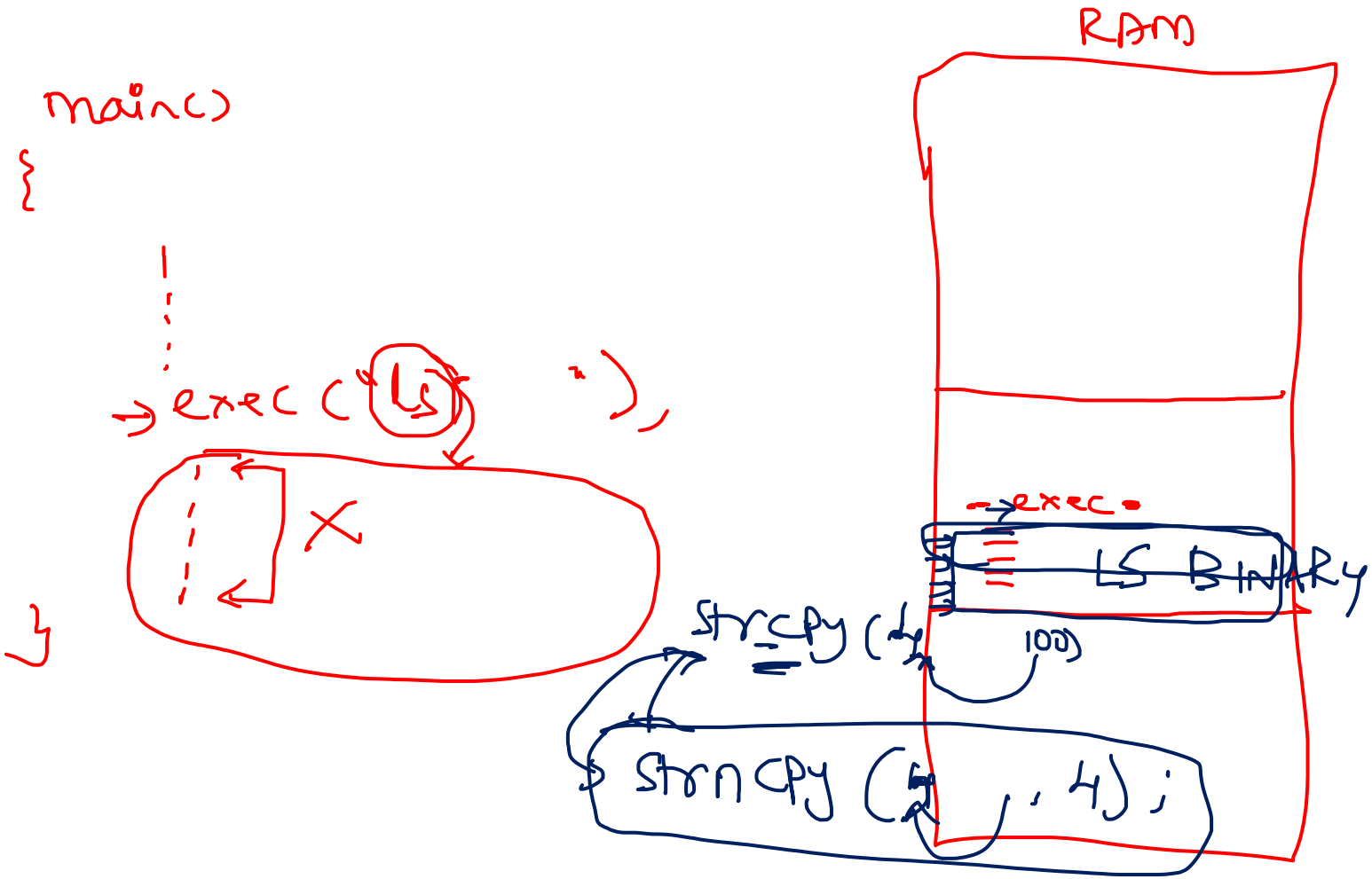



```
if(pid -> )
{

}
else
{

}
```

exec



[ ]

execv

execvp

execve

execl

execlp

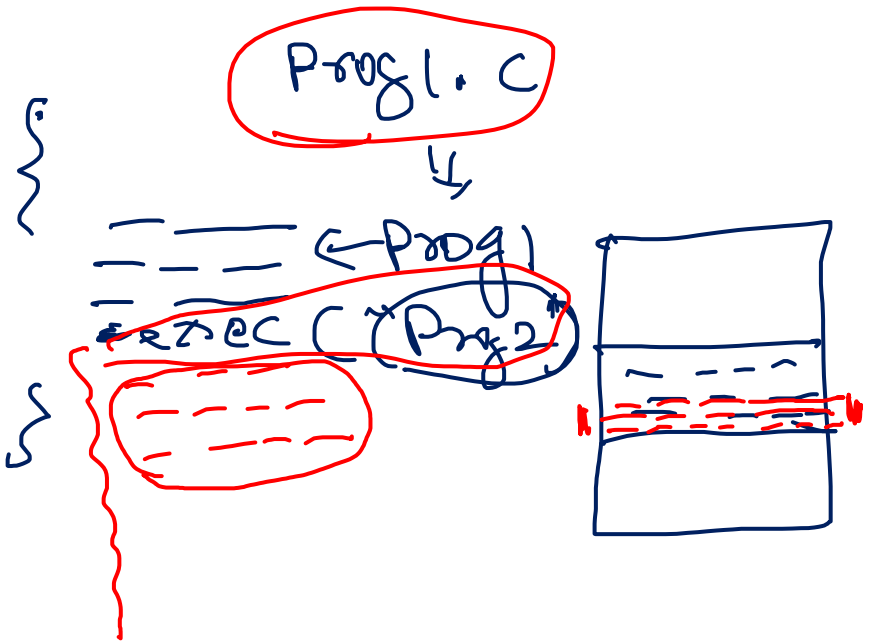
execle

prog2

ps -ax ; ...

exec ( ) ps -ax

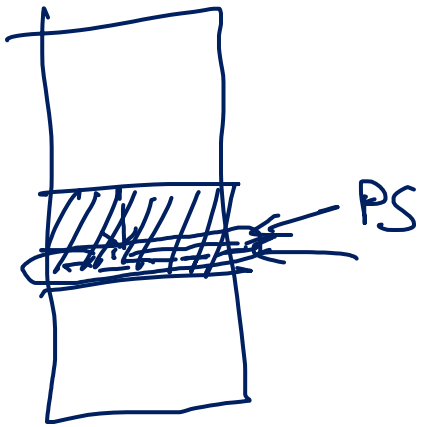
→ execl ("/bin/ps", "ps", "ax", 0);



Prog2.c

↓

prog2

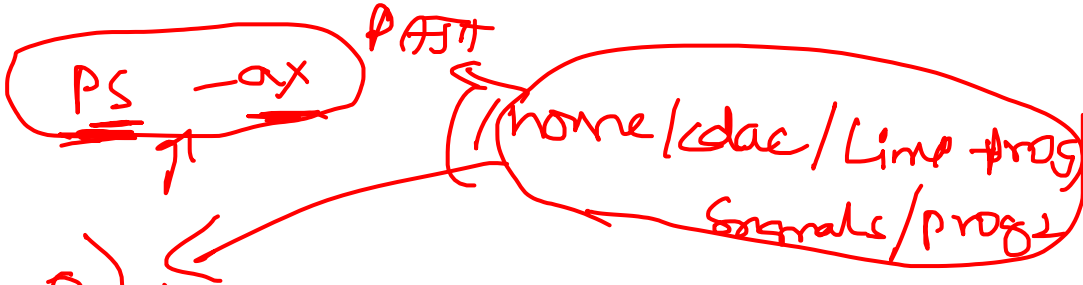


exec

↳ execv, execvp, execve..

↳ exec, execp, execle

PATH



execl ("/bin/ps", "ps", "ax", 0);

execlp ("ps", "ps", "ax", 0);

Path

, "Prog1", 0);