

task is done (success)  
task is not done  
(failure)

want that's called  
exit status of  
1 server.

is logout then  
exit status of  
1 server.

sometime all  
are zombie.

**\$ main \$ exit :-** {("In abc\_main.out") writing}

<u>Exit()</u>	<u>- Exit()</u>
1) It is a library function.	1) It is a System Call.
2) Prototype :-	2) Prototype :-
Void exit (int status)	Void - exit (int status)
3) The exit() function cause normal process termination "immediately".	3) The function - exit() terminates the process
4) all function registered with atexit(3) and on exit(3) are called in the reverse order of their registration.	4)

**\$ Man the atexit**  
↳ defining what  
is doing - how to define  
atexit :- registered a function to be called a normal  
process termination.  
Synopsis :- #include <stdlib.h>  
{  
    void (\*exit\_fn)(void);  
    void (\*atexit\_fn)(void);  
    void (\*atexit\_fn)(void);  
};  
Program :- If program :-  
    {  
        # include <stdlib.h>  
        # include <stdio.h>  
        void abc (void)  
        {  
            printf ("In abc\n");  
        }  
        main()  
    }

pass to wait  
then exit

for

```

Pointf ("In main----\n");
atexit (abc);
pointf ("hai---\n");
sleep (10);
exit (0);
}

```

### Output :-

```

In main----
hai---
(after 10 Second)
In abc -557

```

→ Before terminate by exit() function , it is check the is registered function is there or not after it is terminate .

→ If is there it is called . it is in a reverse order .

### \* Program :- exit()

```

#include <stdio.h>
#include <stdlib.h>
Void abc (Void)
{
    Pointf ("In abc----\n");
    atexit (abc);
    pointf ("hai---\n");
    sleep (10);
    exit (0);
}

In main ---
Vector @ubuntu :~ visbe3|
Linux.

Pointf ("In main----\n");
    → hai is not print bcoz of Hai --- buffer is not flush and - exit() is terminate the process that why Hai --- is not print after 10 second .
}

```

→ If exit(0) ; is used then After 10 second Hai is painted and also registered function also printed .

## \$ main Wait :-

process to change the state . Blab on  
**Synopsis :-** #include <sys/types.h>  
# include <sys/wait.h>  
  
**Pid\_t wait (int \* status);**  
int waitpid (idtype , pid , int \* status , int  
options );  
int waitpid ( idtype , pid , siginfo\_t  
infoop , int options );

## \* Program :-

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
main()
{
    if (fork() == 0)
    {
        printf ("In child\n");
        sleep(10);
        printf ("In child after sleep\n");
        exit(1);
    }
    else
    {
        int s;
        if (wait(&s) == -1)
            perror ("Child process error\n");
        else
        {
            if (s & WIFEXITED(s))
                printf ("Child terminated with code %d\n",
                    WTERMSIG(s));
            else
                printf ("Child still alive\n");
        }
    }
}
```

Second)  
check their  
then it is  
use

Linux .  
ut booz  
is not  
xit(0) is  
process  
after  
then  
listed

wait(0);  
printf (" In parent after sleep\n");  
while(1);  
}

Output :- In parent ---

In child - sleep soft sleep of second

In child often sleep - ~~sleep~~ #

In parent often sleep - ~~sleep~~ - 256

### \* Few point for Wait() :-

- ① we used the `Wait()` System Call in a parent.
- ② `Wait()` will block the parent process till the completion of the child.

③ upon successful waiting `Wait()` return the Pid of the child .

④ if there is not child process is then `Wait()` is failed it return -1 .

⑤ `Wait()` function can be used in a two different way :-

1) `Wait(as);`

2) `Wait(o);`

<code>Wait(as);</code>	<code>Wait(o);</code>
<code>wait</code> is waiting for the child process termination and also interested to collect exit status .	parent is waiting for child process termination but not interested to collect the exit status .

### \* Program :-

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
```

### main()

```
if (fork() == 0)
{
    printf ("In child ---\n", getpid());
    sleep(10);
    printf ("In child after sleep ---\n", getpid());
    exit();
}
```

### Completion

end of

is

#include

```
else
{
    int sign;
    printf (" In parent ---\n");
    x = wait (&s);
    printf (" In parent after sleep ---\n", s);
    x = wait (0);
    while (1) j =
```

}

}

Output :-

In child --- 1993

In child after sleep ---

In parent after sleep --- 3 = 256 x = 1993

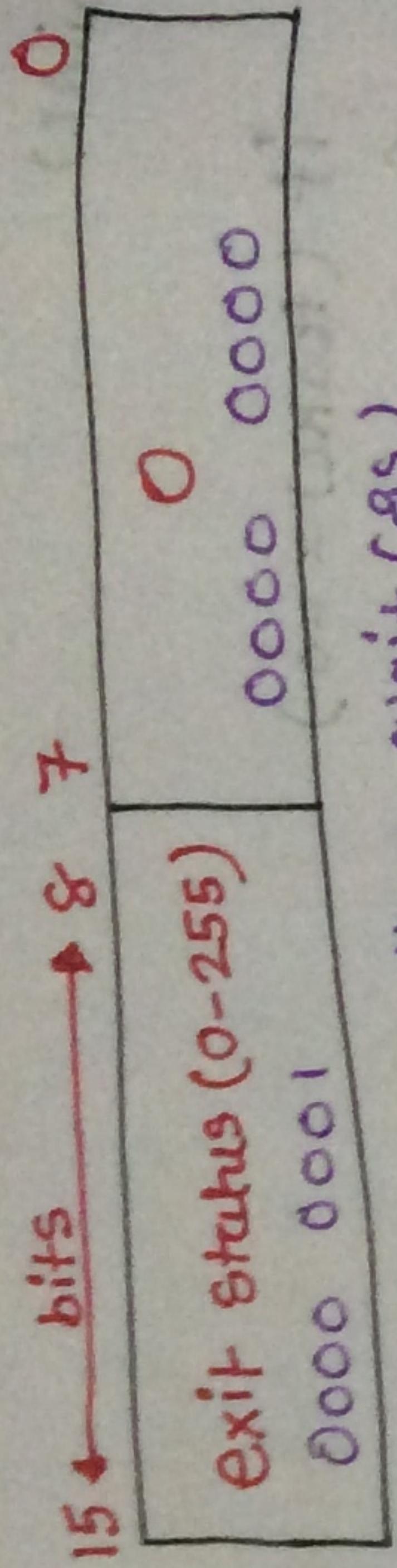
x = -1

\* disadvantage of Wait() :-

- ① Parent Process is Block. waiting to sleep.
- ② Loosing the Job Concurrency.
- ③ Concurrency is not achieve.

\* Advantage of Wait() :-

- ① Child process is remove from Zombie.



mammal avian tetromination

Then  $\theta_{\text{eff}}$   
exit(1) ---  
them want  
SAR and a  
good

```

graph TD
    A["terminated (C!)"] --> B["core dumped"]
    A --> C["child process exit status"]
    B --> D["highly hyphenated normally"]
    C --> E["locally bytes is"]

```

The diagram illustrates the decomposition of the termination signal  $C_! = 0$  into its components:

- terminated ( $C_!$ )** is composed of:
  - core dumped**
  - child process exit status**
- core dumped** is composed of:
  - highly hyphenated normally**
- child process exit status** is composed of:
  - locally bytes is**

Stop by Signal  
addition

Continued by Signal  
with picc)

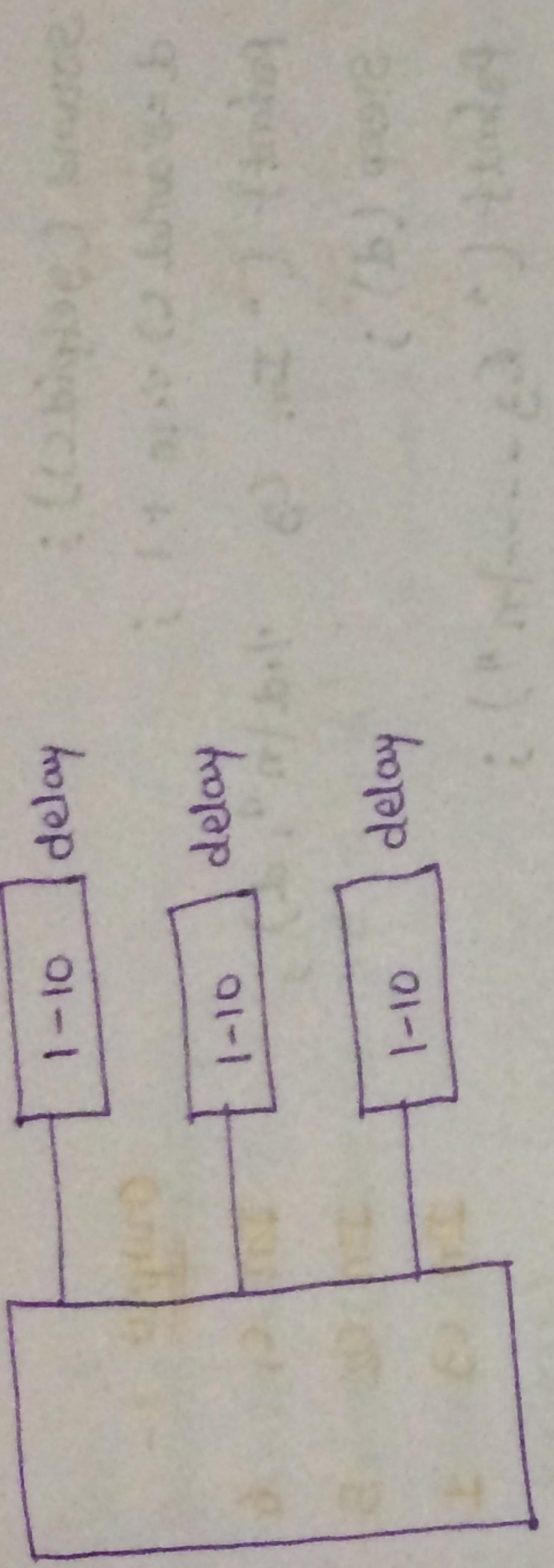
exit(1)  
00000001  
= 256

Assignment  
Example 3  
outline or program to create a child  
process

from a pavement and in each child's  
hand a small bouquet of flowers.  
The young damsel delivered him  
to his mother who had been waiting  
for him at the gate.

① Article a payment Code in such a way that it has to await  
to second or to measure.

also have to pay attention  
to the following points  
in order to make the  
child complete his  
completing the  
3 subjects also  
complete his  
child completion.



Copy	Signature
KOL	KOL

Singer  
R. H.

```
#include <stdio.h>
```

```

if (fork() == 0)
{
    int d;
    srand (getpid());
    d = rand () * 10 + 1;
    printf ("In c1\n", d);
    sleep (d);
    printf ("\n", d);
}

else
{
    if (fork() == 0)
    {
        // child 2
        int d;
        srand (getpid());
        d = rand () * 10 + 1;
        printf ("In c2 - %d\n", d);
        sleep (d);
        printf ("\n", d);
    }
    else
    {
        if (fork() == 0)
        {
            // child 3
            int d;

```

Example-2 Payment should wait until which child complete campsite first?

ausimng exit status .

(above program modification).  
<Stellib.h>

```
#include <stdlib.h>
#include <stdio.h>
```

mission) it took = 0,

if (fork() == 0)

2 Child 1

imperial  
Liaotung  
Province  
China  
1905

Samuel Geppidus;

Sleep (d) ;  
Sleep (p) ;  
Sleep (m, d) ;

exit(1);  
}

```
if (forsk(c) == 0)      bldb ==
```