```
    t1 = time (0);
    printf (" %.s\n", ctime (&t1));
    sleep(1);
    system (" clear ");
}
```

output Tue Nov 17 03:21:56 2015

$ man ctime :-

```
#include <time.h>

char * ctime (const time_t *timep);

char * asctime (const struct tm *tm);
```

Note :-

```
time_t t1,t2;
t1 = time (0)  // not interest to collecting the time.
time (&t1)  // no need to collecting return value.
```

① st_atime :- (last access time)

If any process trying to open the file only for reading purpose then atime is changed.

② st_mtime :- (last modification time)

If any program modifying the content of the file and then m-time is changed.

③ st_ctime :- (last status change time)

If some of the member of inode entry changes then c-time is changed.

Some of member means

- no. of links change.
- no. of size change.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
main(int argc, char **argv)
{
    Struct  Stat V;
    if (argc != 2)
    {
        Printf ("Usage : ./a.out fname\n");
        return ;
    }
    if (Stat (argv[1], &V) < 0)
    {
        Perror ("Stat");
        return ;
    }
    Printf (" atime = %u\n", V.St_atime);
    Printf (" mtime = %u\n", V.St_mtime);
    Printf (" Ctime = %u\n", V.St_Ctime);
}
```

Note :- If we try to open a file in cat command the file open in read mode only.

- If suppose using Vi editor the file may be open in rt mode.

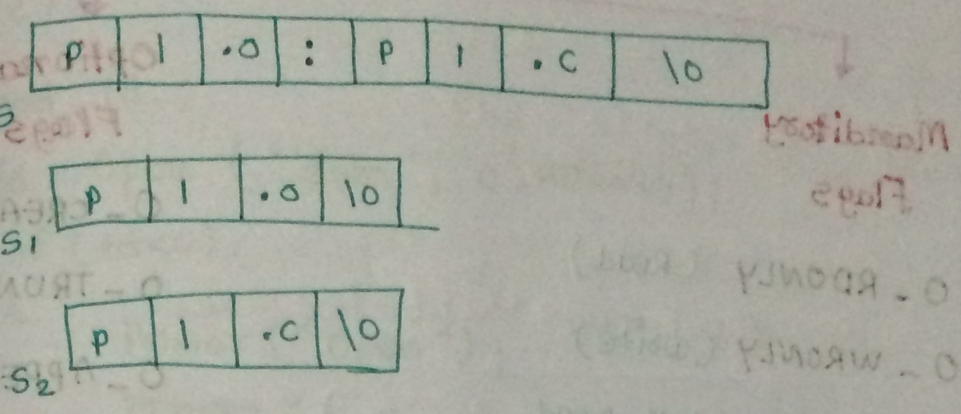
Command :- Cat > filename ⟶ (for Creat).

Cat >> filename ⟶ (for edit).

Design a Oam * make Utility programe.

1) make file

a)



File handling Using System Calls() :-

1) open
2) read
3) write
4) dup
5) dup2
6) Creat
7) fcntl
8) close
9) lseek
⋮

① $ man 2 open :-

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open (const char *pathname, int flags);

int open (const char *pathname, int flags, mode_t mode);

int Creat (const char *pathname, mode_t mode);
```

Flag's

Manditory Flags

→ O_RDONLY (Read)

O_WRONLY (write)

O_RDWR (both read & write)

Optional Flags

O_CREAT

O_TRUNC (truncate data)

O_APPEND (append data)

⋮

✱ File discriptor :-

It is a small non negative integer. which is useful for doing operation on a file.

Vi header.h

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
```

file discriptor is not a structure

- it is a one type of the token number (refrence number).

① program

```
#include " header.h "
main()
{   int fd ;
    fd = Open (" data ", O_RDONLY);
     if (fd<0)
     {
          Perror (" Open ");
          return ;
     }
Printf (" fd = %d\n", fd);
}
```

② program :-

```
#include " header.h "
main()
{
   int fd ;
   fd = Open (" data ", O_WRONLY | O_TRUNC | O_CREAT,
                                          0644);
   if (fd<0)
   {
          Perror (" open ");
          return ;
     }
Printf (" fd = %d\n", fd);
}
```

③ program :-

```
#include " header.h "
main()
{
int fd ;
 fd = open (" data", O_WRONLY | O_APPEND | O_CREAT, 0644);
```

```
    if (fd<0)
    {
        perror ("open");
        return;
    }
    Printf ("fd = %d \n", fd);
}
```

Most important Formula.

① `fp = fopen ("data", "r");`

`fd = open ("data", O_RDONLY);`

② `fp = fopen ("data", "w");`

`fd = open ("data", O_WRONLY | O_TRUNC | O_CREAT, 0644);`

③ `fp = fopen ("data", "a");`

`fd = Open ("data", O_WRONLY | O_APPEND | O_CREAT, 0644);`

① `Vi /Proc /Pid /limits`

② `ls /Proc/1816/fd`

③ `ls -l /Proc/1816/fd`

① $ man

read → re

#incl

ssize_t re

⇒ read() at

discriptor f

success

failure

⇒ Void * r

Pass it.

include "h

main()

{

int fd

fd = op

if (fd

{

Pe

re

}

Printf ("

rea

Printf ("

}

⇒ Read()

⇒ So afte

Program :

① $ man read

read → read from a file discriptor.

```
#include <unistd.h>

ssize_t read (int fd, Void *buf, size_t count);
```

→ read() attempts to read upto count bytes from file discriptor fd into the buffer starting at buf.

success → no. of bytes read is returned.

failure → -1 is return and errno is set.

→ Void * means any type of the address we can pass it.

```
#include "header.h"
main()
{
    int fd;
    fd = open ("data", O_RDONLY);
    if (fd<0)
    {
        Perror ("open");
        return;
    }
Printf ("fd = %d\n", fd);
    read (fd, a, 5);
Printf ("Data = %s\n", a);
}
```

→ Read() function is not put the '\0' at the end.

→ So after 5 bytes Junks data is come.

Program :-

```c
#include " header.h "
main()
{
    // char a[20] = { } ;

    char a[20] ;
    int fd ;
        fd = open ("data", O_RDONLY) ;
        if (fd<0)
        {
            Perror (" Open") ;
            return ;
        }
    Printf (" fd = %d|n", fd) ;
    bzero (a, sizeof (a)) ;
        read ( fd, a, 5) ;
    Printf (" Data = %s|n ", a) ;
}
```

2 function

bzero()

memset()

② $ man bzero

bzero → write zero Valued bytes.

```c
#include <strings.h>
```

```c
Void bzero (Void *s , size_t n) ;
```

Return :- None..

⇒ The bzero() function sets the first n bytes of the byte area starting at to zero. (bytes containing '\0'.)..

③ $

memset

#i

Void *

return Val

The

memory

Discription

⇒ The

memor

byte

⇒ In

any

#includ

main()

{

char

int

fd =

if

{

}

Print

ret =

Print

Print