Date : 1/11/23

Linux File System

Guide:Ramesh

FHS(**File system Hierarchy Standard**)->It has /bin files.

bin,user,/var,/home.....

->bin directories in linux are dedicated areas for storing executable files.
->sbin is system binary -> It will storing admin details.
->mutex is locking mechanism.
->Semaphore is signaling mechanism.
->wait() is an inbuilt command in the linux shell.
->signal is generated by a process via kernel.
->Binary semaphore has two values:0 & 1
    0->not available resource
    1->available resource
->counting semaphore is counting the running resources.
->/home->user directory
->/temp ->it will store the data temporarly
->/etc ->used to modify while running the program.

TASK 1

**Write a C function that takes an array of integers and the size of the array as input. The function
should return the second largest element in the array. If there is no second largest element, the function should return an appropriate message or handle the situation accordingly ?**

```
#include<stdio.h>
int main()
{
        int i,high=0,sh=arr[0],n,arr[100];
        scanf("%d",&n);
        for(i=0;i<n;i++)
                scanf("%d",&arr[i]);
        for(i=0;i<arr[i];i++)
        {
             if(high<arr[i])
            {
                sh=high;
                high=arr[i];
            }
        else if(sh<arr[i])
        {
                sh=arr[i];
        }
for(i=0;i<arr[i];i++)
        printf("%d",arr[i]);
}
```

TASK 2

**Given an array of integers, write a C function to find the maximum and minimum elements in the
array and return them using pointers ?**

```
 #include<stdio.h>
void min_max(int *,int *,int);
int main()
{
        int arr[100],size,min,max,i;
        scanf("%d",&size);
        for(i=0;i<size;i++)
                scanf("%d",&arr[i]);
        max=arr[0];
        min=arr[0];
        minmx(arr,&min,&max,size);
}
void min_max(int *ptr,int *min,int *max,int n)
{
        for(int i=0;i<n;i++)
        {
        if(*max<*ptr)
        {
                *max=*ptr;
        }
        if(*min>*ptr)
        {
                min=ptr;
        }
        ptr++;
        }
}
```

TASK 3

Write a c program that takes an array of integers as input & finds the maximum product of two integers in the array.

```
#include<stdio.h>

int main()

{

        int a[]={1,4,2,3,6,7,0},I,max=arr[0],size,sec=0;

        size = sizeof(arr)/sizeof(arr[0]);

        {

              if(arr[i]>max)
```

```
                {
                   Sec=max;
                   Max=arr[i];
                  }
                  if(arr[i]>sec && arr[i]!=max)
                         Sec=arr[i];
          }
          Printf("Product of max integers=%d * %d = %d\n",max,sec,max*sec);
}
```

Process state

Guide: Sandeep

->created
->Ready
->Running
->waiting
->zombie

->fork():- It creates child process to the parent process.
->vfork():- It creates new process .
->exec():- It is used to execute a command from the bash itself.

Commands :-

```
 ->PS
 ->TOP
 ->cat/proc
 ->PS/aux
        |
     all user user/owner list of process
 ->chmod: it allow the permissions.
```

02/11/23

Guide:Ramesh

Static library:

It is created copy of all the files(object files)which may execute program.

TASK 4

Write a code for creation of staic library ?

Create file:

lib_mylib.c

```
#include<stdio.h>
void fun(void)
{
            printf("fun() static library");
}



 #include<lib_mylib.h>
int main()
 {
    fun();
  }

fun();
```

TASK 5

Input ----Hello world

Output – dlrow

```
#include<stdio.h>
#include<string.h>
int main()
{
             char s[]="Hello world",s2[20], temp;
             int i=0,len;
             len =strlen(s)-1;
             while(len)
             {
                              temp=s[len];
                             s2[i]=temp;
                             len--;
                             i++;
                            if(s[len]==32)
                            break;
            }
            Printf("%s",s2);
}
```

TASK 6

Write a c program to reverse a given string and print that string ?

```
#include<stdio.h>
#include<string.h>
int main()
{
    char str[20];
    printf("Enter String: ");
    gets(str);
    printf("Entered string is: %s\n",str);
    int start=0,end=strlen(str)-1;
    while(start<end)
    {
        char temp=str[start];
        str[start]=str[end];
        str[end]=temp;
        start++;
        end--;
    }
    printf("Reversed string is: %s",str);
    return 0;
}
```

Guide:Ramesh **03/11/2023**

Static	Dynamic
->It happens last step of the compailation process. After the program is placed in the memory.	->shared libraries are added during the Process when executable files and libraries are added to the memory.
->performed by linker	->performed by OS
->much bigger ,bcz external programs are built in the executable file.	->much smaller,bcz there is only one copy of the dynamic library that is kept in memory.
->.a file	->.so file

TASK 7

Write a c program using file handling functions fopen(),fread(),fwrite,fclose() ?

```
#include<stdio.h>
#include>stdlib.h>
char *read(FILE *,char *);
void write(FILE *,char *);
long long int n;
int main()
{
            FILE *src,*dest;
            char *buff;
            buff=read(src,buff);
            write(dest,buff);
}
char *read(FILE *fp,char *buff)
{
            fp=fopen("filename","r");
            if(fp==NULL)
```

```
        {
                printf(“FILE not found”);
        }
        fseek(fp,0,2);
        n=ftell(fp);
        rewind(fp);
        buff=calloc(1,n);
        fread(buff,n,1,fp);
        fclose(fp);
}
```

Guide:Ramesh **06/11/23**

TASK 8

Write a C program that takes a string as input, prints the length of the string, and then prints the string in reverse ?

```
#include<stdio.h>

#include<string.h>

int main(void)

{

char mystrg[60];

int leng, g;

printf("Program in C for reversing a given string \n ");

printf("Please insert the string you want to reverse: ");

scanf( "%s", mystrg );

leng = strlen(mystrg);

for(g = leng - 1; g >= 0; g--)

{

printf("%c", mystrg[g]);

}

printf(“%d”,leng);

return 0;

}
```

Guide:Ramesh **07/11/23**

Task 9

Write a C program that reads a text file named "input.txt" and copies its content to another text file named "output.txt" using file handling functions and error handling ?

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
FILE *src,*dest;
int ch;
src=fopen(“input.txt”,”r”);
if(src==NULL)
{
printf(“no such file”);
}
dest=(“output.txt”,”w”);
while((ch=fgetc(src))!=EOF)
{
fputc(ch,dest);
}
fclose(src);
fclose(dest);
}
```

TASK 10

Write a C program that reads a CSV (Comma Separated Values) file named "data.csv" containing numbers and calculates the sum and average of the numbers using file handling functions ?

```
#include<stdio.h>
int main()
{
        FILE *src,*dest;
        int n,sum=0,cnt=0;
        src=fopen("data.csv","r");
        if(src==NULL)
        printf("File not found");
        dest=fopen("output.txt","w");
        while(fscanf(src,"%d,",&n)==1)
        {
                sum+=n;
                cnt++;
        }
        float avg1 =(sum/cnt);
        fclose(src);
        fprintf(dest,"%d %f",sum,avg1);
        fclose(dest);

}
```

TASK 11 08/11/23

Write a C program that reads a CSV (Comma Separated Values) file named "data.csv" containing numbers and calculates the minimum,maximum values ,sum and average of the numbers using file handling functions ?

```
#include<stdio.h>
void main(){
      FILE *src,*des;
      src=fopen("existing.c","r");
      if(src==NULL){
            printf("the file is empty");
      }
```

```
        des=fopen("res.c","w");
        int a[30],min,max,i=0,cnt=0,n,j;
        while(fscanf(src,"%d",&n)==1)

        {
             a[i]=n;
             i++;
            cnt++;
        }
                min=max=a[0];
          for(j=1;j<cnt;j++)
         {
          if(a[j]>max)
          max=a[j];
        if(a[j]<min)
                 min=a[j];
         }
        fclose(src);
        fprintf(des,"%d %d",min,max);
        fclose(des);
}
```

Guide:Ramesh **09/11/23**

TASK 12

Write a program taking a string as input in run time and write that string in to ram.txt file ?

```
#include<stdio.h>
int main()
{
             FILE *fp;
             fp = fopen("ram.txt", "w");
             char str[50];
             gets(str);
             for(int i = 0; str[i]; i++)
             fputc(str[i],fp);
            fclose(fp);
}
```

TASK 13

Write a C program that appends a string to a file named 'output.txt'. Implement a function called 'appendToFile' that takes a string as input and appends the string to the file 'output.txt'. Ensure that the file is opened in append mode. Test the function with different

string inputs to verify that the appending operation functions as expected without encountering any errors.

```
#include<stdio.h>
void AppendToFile(char *);
int main()
{
        char str[100];
        AppendToFile(str);
}
void AppendToFile(char *ptr)
{
        FILE *fp;
        fp=fopen("output.txt","r");
        fgets(ptr,100,fp);
        fp=fopen("output.txt","w");
        fwrite(ptr,6,1,fp);
}
```

Guide:Sandeep

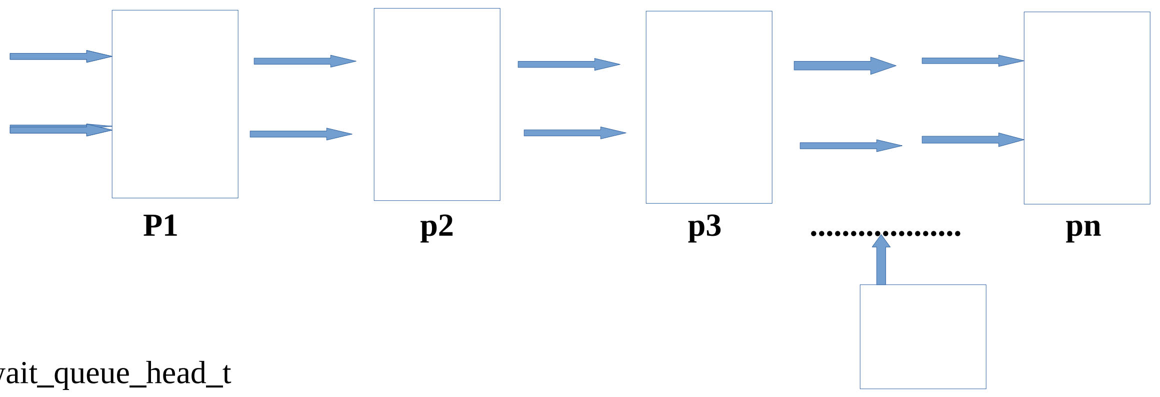
PROCESS

Hearderfile	**Sourcefile**
->include/linux/sched.h	->kernel/sched.c
->include/linux/wait.h	->kernel/signal.c
->include/asm_i386/system.h	->kernel/fork.c
->include/arm/system.h	->arch/i386/kernel/entry.s
	->arch/arm/kernel/entry.s

```
                    multitask
                        |
                      PID    3G   fork
                     USER         exec
   by using   ----|----------------------------
   system calls
                     kernel   4G
                                  vfork()
                    Hardware      vexec
```

=>Linux Process Management :

->Queue for all Tasks
->Queue for all running Tasks
->Any task temporarily are blocker(round robin)



->wait_queue_head_t
->wait_queue_t

=>Fork(system call) Internally

->Libc fork()

system_call(arch/i386/kernel/entry.s)

sys_clone(arch/i386/kernel/process.c)

Guide:Ramesh **10/11/23**

TASK 14

Write a C program that reads a binary file containing a complex data structure, processes each record, and selectively writes the modified records to an output file. The challenge is to manage memory efficiently, handle various data types within the file, and implement read, write, and append mode operations without causing memory leaks or performance issues, even when dealing with large files.

```
#include<stdio.h>
struct student
{
        char name[20];
        int roll;
}s;

int main()
{
        FILE *fp,*src,*dest;
        char *buff=NULL,ch;
        src=fopen("aa.c","r");
```

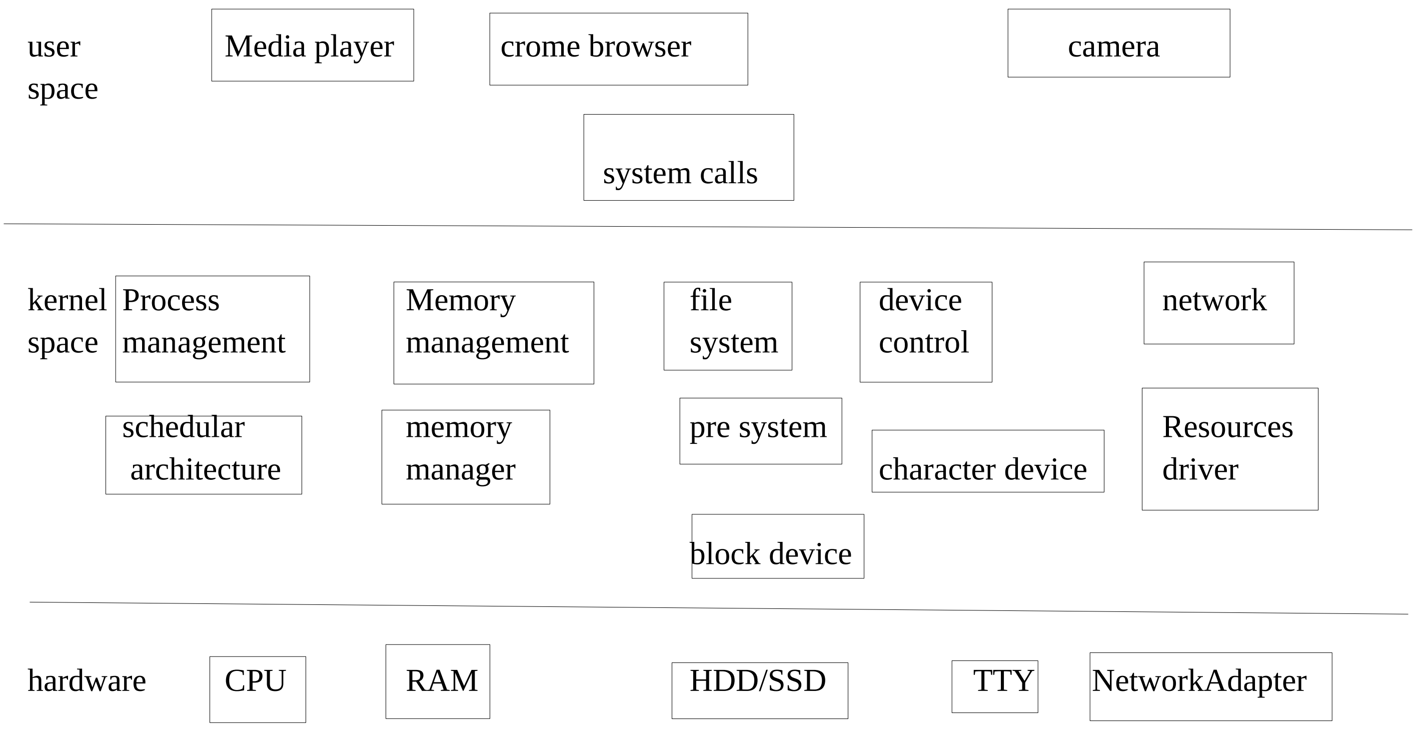
```
	if(src==NULL)
		printf("File not found");
	dest=fopen("bb.c","w");
	while((ch=getc(src))!=EOF)
	{
	fputc(ch,dest);
	}
	fp=fopen("bb.c","a");
	scanf("%s %d",s.name,&s.roll);
	fprintf(fp,"%s %d",s.name,s.roll);
	fclose(src);
	fclose(dest);
	fclose(fp);

}
```

TASK 15 14/11/23

Write a c program to enter data after given input (PTG welcomes you all) using r+ mode in file handling ?

```
#include<stdio.h>
int main()
{
FILE * fp=NULL;
char ch;
fp=fopen("sample_file.txt","r+");
if(fp==NULL)
{
perror("message:file not found");
return 0;
}
while((ch=getc(fp))!=EOF)
{
	printf("%c",ch);
}
char str[100];
printf("enter the string to write in the file\n");
scanf("%[^\n]s",str);
fputs(str,fp);
fclose(fp);
}
```

LDD (Linux Device Driver)



Guide:Ramesh **15/11/23**

TASK 16

Write a c program using fseek function in file handling ?

```
#include <stdio.h>
int main()
{
        FILE* fp;
        fp = fopen("test.txt", "r");

        fseek(fp, 0, SEEK_END);
        printf("%ld", ftell(fp));

        return 0;
}
```

TASK 17

Write a c program to find character count, line count in a file ?

```
#include<stdio.h>
int main()
```

```
{
        FILE *fp;
        char str[20],ch;
        fp=fopen("c.c","a+");
        if(fp==NULL)
        printf("File not found");
        fputs("esh",fp-1);
        fclose(fp);
        fp=fopen("c.c","r");
        while((ch=fgetc(fp))!=EOF)
        {
                if(ch=='\n')
                        continue;
                printf("%c",ch);
        }
        fclose(fp);
}
```

Guide:Ramesh **16/11/2023**

TASK 18

Write a program using error handling in c (ferror and feof) ?

feof:

```
#include<stdio.h>
int main()
{
     FILE *fp;
     char ch;
     fp=fopen("2.c","r");
     if(fp==NULL)
          printf("File not found");
     else
     {
          fseek(fp,0,2);
          while((ch=fgetc(fp))!=EOF)
          {
               printf("%c",ch);
          }
               if(feof(fp))
               {
                    printf("Feof error");
               }

     }
     fclose(fp);
}
```

ferror:

```
#include<stdio.h>
int main()
{
	FILE *fp;
	char buff[100];
	fp=fopen("1.c","r");
	fwrite(buff,100,1,fp);
	puts(buff);
	if(ferror(fp))
		printf("Invalid");

	fclose(fp);
```

Guide:Ramesh **Shell Scripting** **20/11/23**

Task 19

Delete a line and insert a line in file using shell scripting?

/*lines1.sh*/

```
echo -e "1.aa
2.bb
3.cc
4.dd
5.ee
6.ff
7.gg
8.hh
9.ii\r9.pppp
10.jj"
```

Task -20

Write a simple shell script using echo and some commands ?

```
#!/bin/bash
sum=$((1+2))
echo "sum is:$sum"
```

Guide:Ramesh 21/11/23

Task 21

Write a simple shell script that reads each line from data.txt and prints "Fruit: [fruit name]" for each line.

1. **Write a shell script to accomplish this task**
2. **Ensure that the script handles potential errors such as the absence of data.txt .**

```
        /*fruits.sh*/
    #!/bash/bin/
    while read ch
    do
            echo "$ch"
    done < data.txt

/*data.txt*/
"fruit: apple"
"fruit: banana"
```

Guide:Ramesh 22/11/23

Task – 22

Write a simple shell script to copy a file properties and content in another file without changing timestamp ?

```
read -p "Enter the file name:" file
v=$(stat $file)
filename="data1.txt"
while IFS= read -r line
do
        echo "$line"
done < Prop.sh > data1.txt

if [ -e $file ]
then
        echo $v>>data1.txt
else
        echo "No file"
fi
```

Commands:

nano – To give permission to file.
vi - To create a file or to open a file.
head--It gives first 10 lines of any given file.
            syntax: head filename.txt
tail--It gives last 10 lines of given file.
             syntax: tail filename.txt
|tr--used to replace a character.
            eg: a b c d |tr   c e-----input
            a b e d-------output
--a Shell scripting using functions.

```
i) a()
 {
  echo $1
  echo $2
 }
   a b c
  output---b
             c
ii)pavan()
{
             echo $1
           echo $2
 }
 pavan kumar kotha
```

To append a word to a file command:-

```
sed –i ‘s/$’pavan/’filename.sh
----> echo “a b c d”|tr c e
            output --> a b e d
--->echo “a b c d”|cut –d’ ‘-f3
            output ----> c
--->echo “a b c d” |cut –d’ ‘-f4
       output ----> d
--->echo “a b c d” |sed‘s/b/p’
       output ----> a p c d
--->echo “a b c d”|awk‘{print $2}’
       output ----> b
--->echo “a b c d”|awk‘{print $0}’
      output ----> a b c d
```

<u>On console:-</u>

```
a=1
echo $a
output --> 1
```

if we give **bash** it will clear.

export a=1
echo $a
bash
echo $a
output --> 1
by giving export a=1 it saves it will not clear even use bash.

Guide: Sandeep

--->First download tarball linux-4.19.299.xz
link---- https://github.com/torvalds/llinux

---->PTG_LLD directory

mkdir GIT_LDD
cd GIT_LDD

To download git extension files
git clone https://github.com/torvalds/linux.git
specific kernel version-------linux-4.19.299
command ---- ll---> it shows all hidden files dot extension available
 ----> it contain.git/ ---file
cd LDD_TRAINING/linux/.git/
--->git branch
*master
git branch –r -->list of origin
git branch –a -->show all branch
for git configuration:--
git init ------ to install git
git config -- global user.name “name”
git config – global user.email   “name@rampgroup.com”
cd drivers/i2c/
mkdir ptg_i2c
cd ptg_i2c/
vi i2c_ptg.c

Guide:Ramesh

27/11/23

Linux Commands:--

diff- Find the difference between two files.

dirname- extracting the directory portion from a given path or filename.

dmesg- print or control

fdisk- change partition table.

free- display the amount of free and system memory usage.

grep – search for pattern in each file or standard output.

su –change the user id or become root.

Task 23

Write a program on how to create a child process and kill the parent process?

```
#include<stdio.h>
#include<unistd.h>
int main()
        int ret;
        ret=fork();
        if(ret==-1)
                printf("error");
        else if(ret==0)
        {
                printf("Child process\n");
                printf("process id is %d\n",getpid());
                kill(getppid(),9);
        }

        else
        {
                sleep(30);
                printf("parent process id is %d\n",getppid());
        }

}
```

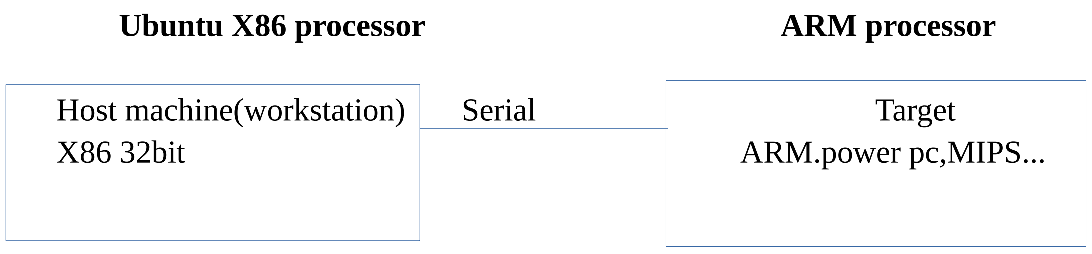
Guide:Sandeep **Embedded Linux** **28/11/23**

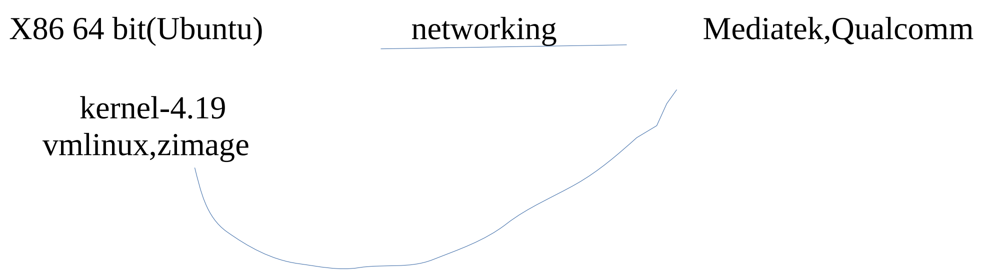
Linux Distribution:

ubuntu,debian,fedora,opensuse,arch linux.

Why Ubuntu:

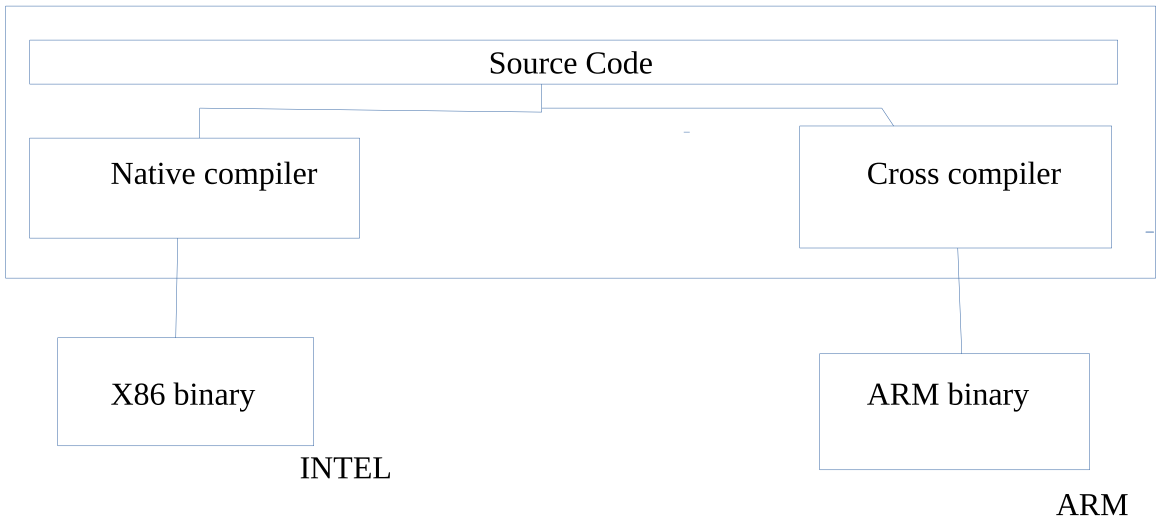
because it is easy to install and easy to use.





->serial line communication program : for this we are using gtkterm,minicom -> for linux ubuntu
->SERIAL DEVICE is typically,ttyUSBx for USB to serial converter.
Ttysx,x=0,1,2,3...for serial port..

<u>Tool Chain :</u> Convert host to Target.
->mostly we are using GNU native tool chain.
->Required for cross compilation ,this will run on your workstation but generate code for your target machine like ARM.



Components of Tool Chain :

->Binutils
->kernel header
->c/c++ libraries
->c/c++ compiler
->GDB debugger(optional)

<u>Sigterm</u>

Kill-15 <pid>

1. Doesn’t kill the child process

2. It will handled ignored blocked

3. Greacefully kil the process

<u>Sigkill</u>

Kill -9 <pid>

1. Kill the child process as well

2. Can’t be handled blocked

3. Kill the process immediatly

Mostly we don't use "sigkill" we use "sigterm"
Eg: fork();
fork();
fork(); **(2^n-1)=2^3-1**
printf("PTG"); **8-1=7**
Output: 7 times PTG will print
1*st*fork() will create child and parrent process
2*nd*fork() wil create 2 child process and 3*rd* fork() will create 4 child process. Total 7 child process.

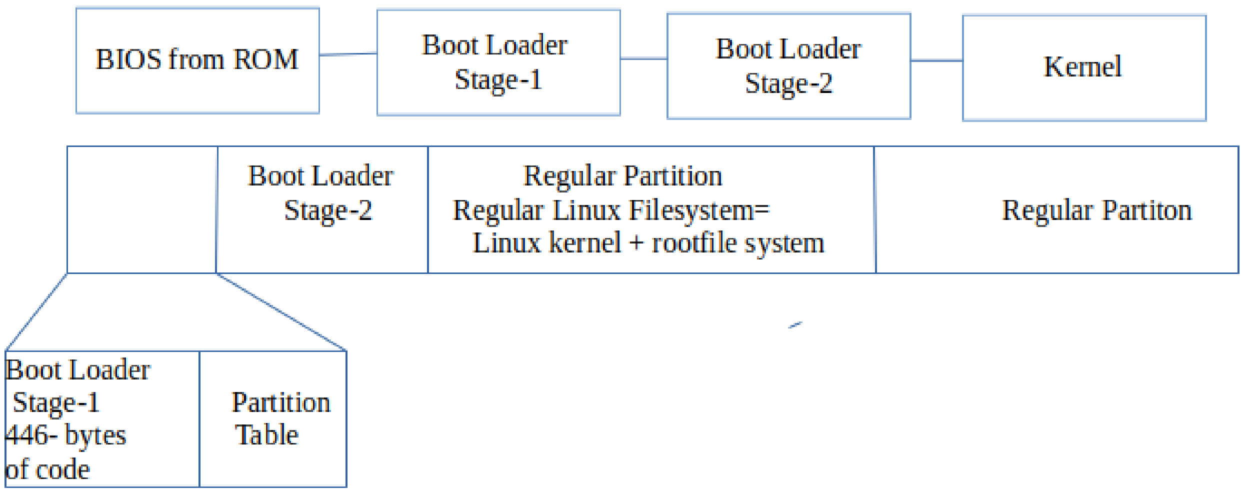
Application Binary Interface (ABI):

<u>Binary Image:</u>
In ARM cortex 64 or 32 two types of ABI's are there
1. EABI--->Floating point argument
2.EABhf

ARM version 4 & 5 doesn't support hf (floating point).

Requirements:CPU, endianness, C library , kernel header, ABI (soft hard float)
GNU is an open source & we are using ARM and ARM tacking licence from GNU & making the tool chain. Linaro is also same like ARM Mostly Qualcom using linaro tool chain.

Boot Loader:
Boot loader is used for kernel initialization.
By using some cammands we can load images and boot loader is like basic hardware initialization.
Loading application binary from one storage to another storage.
Lenovo Machine X86 platform BIOS, support flash memory


BIOS from ROM
Boot Loader Stage-1
Boot Loader Stage-2
Kernel
Boot Loader Stage-2
Regular Partition Regular Linux Filesystem= Linux kernel + rootfile system
Regular Partiton
Boot Loader Stage-1 446- bytes of code
Partition Table

Guide: Sandeep **Embedded Linux** **01/12/23**

BootLoader Processing:

- ➔ C source file code
- ➔ Compiler
  - Eg: arm-none-eabi-gcc
- ➔ Object file
- ➔ Linker
  - Eg:arm-none-eabi-ld
- ➔ Program file
- ➔ programming utility
- ➔ MEDIATEK/QUALCOMM board

ABI Processing :

API

Linux kernel to User space	**Linux kernel Interface**
● System call Interface	->Linux memory manager
● Linux	->TPC manager
	->Network Interface
	->I/O interface

ABI

Linux kernel Interface	**Linux kernel Interface**
Tools will create ABI Image	Binary device driver kernel version 4.19
● Linux "A1"	->DRM kernel version 5.2
● Linux "B1"	->DRM kernel version 5.19
● Linux "C1"	->DRM kernel version 6.2

RAM Code:

- ➢ It is the initial step of the boot process.
- ➢ We cannot change or update.
- ➢ For finding the suitable bootloader like BIOS,UFEI,UBOOT ..loading it and running successfully.
- ➢ We can use NAND/NOR flash memory from USB drive,from SD card,from EMMc.
- ➢ It is initialized externel SRAM,it will load the bootloader into an internal SRAM.
- ➢ RAM Code contains limited size of bootloader,due to SRAM size.
- ➢ Forces the boot process to be split in two stages.
  - • bootloader(small,runs from SRAM,Initializeexternel DRAM).
  - • bootloader(Large ,runs from externel DRAM).

Guide: Sandeep **Embedded Linux** **04/12/23**

Booting on X86 Platform:

BIOS & MBR --->Legacy BIOS Booting.
ACPI -->Advanced Configuration & Power Interface

Two Stage Booting Sequence:

➔ **EMMC**
 1st Stage Boot Loader
 2*nd* Stage Boot Loader
 Rest of the system

➔ **SOC**
 CPU Core
 MMC Controler
 ROM Code
 DDR Controller
 SRAM
 Other Pheripheral

➔ **DDR Memory**

➢ The ROM Code loades the 1*st*stage & excecutes it

➢ The 1st stage bootloader running from SRAM, Configures the DDR controller

➢ The 1st stage bootloader, running from SRAM, loades the 2*nd*stage into external DDR & excecute it.

GRUB-X86: Grand unified bootloader --> from GNU project

- GRUB supports ARM, ARM-64, PISC-V, Power PC,INTEL.....
- Used in linux distribution and it can read many filesystem & it load kernel image modules &provide a MENU & POWERSHELL with various cammands.
- **uname -r** --> to see kernel version
- support ARM,ARM-64,RISC-V,POWER PC,INTEL....

ARM UBOOT:

- In Embedded linux we are using UBOOT.
- UBOOT also like a kernel.
- Supports ARM, ARM64, RISC-V, POWERPC, MIPS etc.... X86 with UEFI software.
- We have to port uboot image to embedded board (NXP, Qualcom) it is an open source.
- For example we are working on NXP IMX6 board ?(manual for IMX6).
- **Deconfig file:** It will help to configure your board file IMX6- deconfig ----- CONFIG-PTG-TOUCH
- Upstream means everyone will use code.

Guide : Ramesh **Process Management**

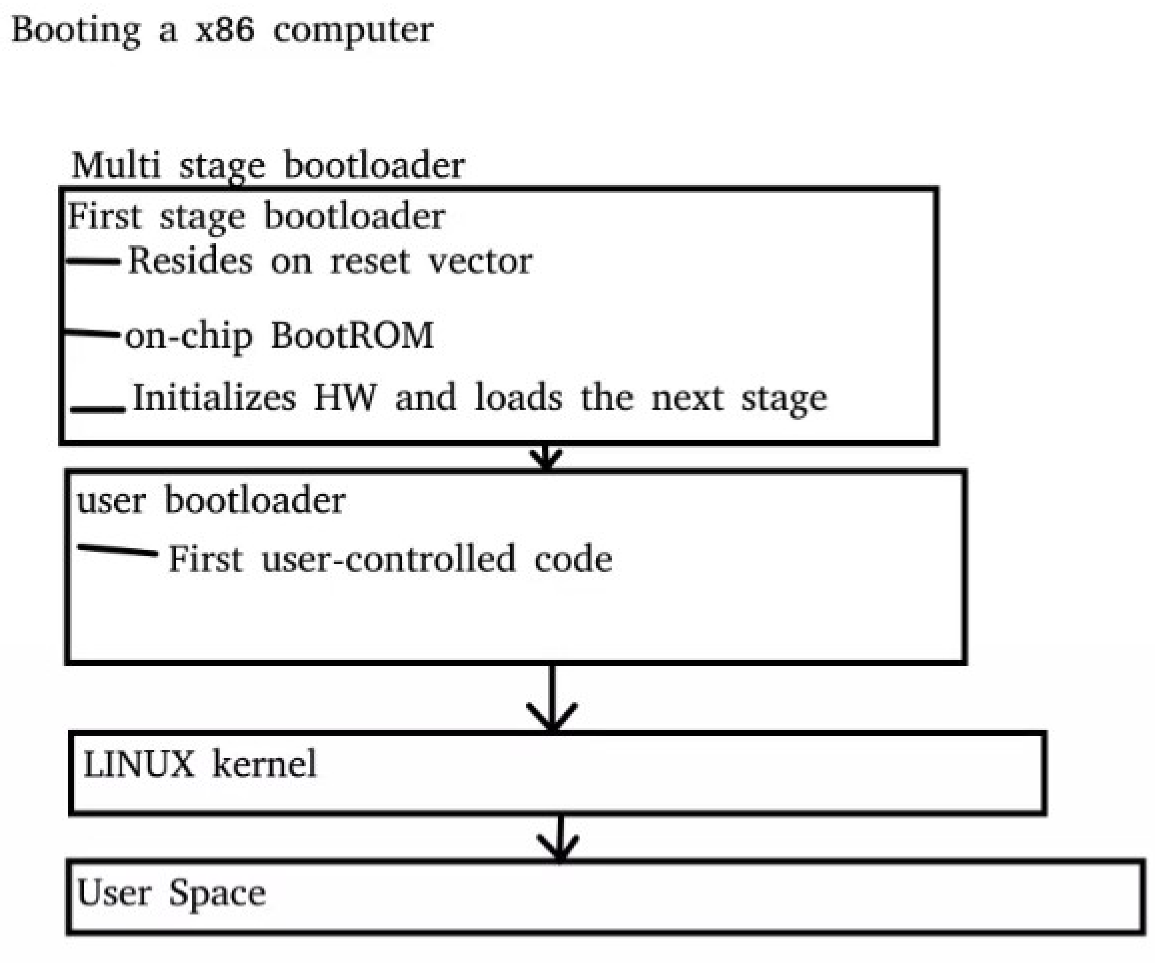
Task :

```
/*Write a C program that uses fork() and pipes to implement inter-process
communication (IPC) between the parent and child processes.*/
#include<stdio.h>
#include<unistd.h>
int main()
{
int p[2];
pipe(p);
if(fork()==0)
{
int i;
for(i=0;i<10;i++)
{
sleep(1);
write(p[1],&i,2);
}
}
else
{
int j;
do{
read(p[0],&j,4);
printf("%d\n",j);
}while(j<10);
}
}
```

Write here with commenting by each line, what it is heap or stack, free heap etc.

```
#include <stdio.h>
//stack segment
#include <stdlib.h> //stack segment
int main() { //stack segment
int a = 10;
//stack segment
int *d = malloc(sizeof(int)); //it allocate memory in heap segment
*d = 40;
//heap segment
free(g);
//free heap segment
free(j);
//free heap segment
free(k);
//free heap segment
int *g = malloc(sizeof(int) * 10);
//allocate in heap segment
printf("a = %d\n", a); //stack segment,intiliazed
printf("b = %d\n", b);
//stack segment,intiliazed
printf("c = %d\n", c); //stack segment,initialized
printf("d = %d\n", *d);
//stack segment,initialized
printf("e = %d\n", e); //stack segment,initialized
printf("f = %d\n", *f);
//stack segment,uninitialized
int e = 50;
//stack segment
04/12/23int *f;
//stack segment
printf("g[0] = %d\n", g[0]); //heap segment,unintialized
printf("g[1] = %d\n", g[1]); //heap segment, unintialized
printf("g[2] = %d\n", g[2]); //heap segment, unintialized
printf("h[0] = %d\n", h[0]); //stack segment, unintialized
printf("h[1] = %d\n", h[1]); //stack segment, unintialized
printf("h[2] = %d\n", h[2]); //stack segment, unintialized
printf("i = %d\n", *i); //stack segment,initialized
printf("j = %d\n", *j); //stack segment,initialized
int h[10];
//text segment
int *i = malloc(sizeof(int));
//heap segment
*i = 60;
//heap segment
int *j = malloc(sizeof(int));
//heap segment
*j = 70
//heap segment
int *k = malloc(sizeof(int) * 5);
//heap segment
int b = 20;
//stack segment
int c = 30;
//stack segment
free(d);
// Freeing heap memory segment
printf("k[0] = %d\n", k[0]); //heap segment, unintialized
printf("k[1] = %d\n", k[1]); //heap segment, unintialized
printf("k[2] = %d\n", k[2]); //heap segment, unintialized
return 0;
//stack segment
}
```

- **TASK:**
-
- **UBOOT directories:**
-
- arch/->Arch or platform related DTS,cpu init and pinmux controller,DRAM,clock.
- Board/->board specific(init, pinmuxing,config),kconfig file specifying,board header file,board file,patha,makefile for board file.
- Config/->all board defconfig.
- Include/->all header
- include/configs/->all board header file.
-



-
- **Booting a ARM Uboot Bootloader :**
-
- First stage bootloader
- UBOOT SPL (secondary program loader)
  ->First user-controlled code
  ->Responsible for additional Hardware initialization.

->Loads UBOOT or linux directory.

-
- UBOOT
  ->Bootloader with interactive shell.
  ->Boot monitor
  ->Debugg Tool
- Kernel space
- User space
- ->Hardware support
- ->Architecture/SOC
- ->X86,MIPS,PowerPc,RISCV
- ->Board : It will support approx 190+ board vendor,1085+ diff board.

Guide:Ramesh **Processing Management** **05/12/23**

TASK:

/*Write a C program that validates command line arguments and prints an error message if invalid arguments are provided.*/

```
#include<stdio.h>
int main(int argc,char *argv[])
{
	if(argc>1)
		printf("error bcz invalid arguments\n");
	return 0;
}
```

/*Write a C program that uses command line arguments to pass configuration options or input data to the program.*/

```
#include<stdio.h>
#include<string.h>
int main(int argc,char *argv[])
{
	int i;
	for(i=0;i<strlen(argv[1]);i++)
	{
		printf("%c",argv[1][i]);
	}
	return 0;
}
```

By using Thread:

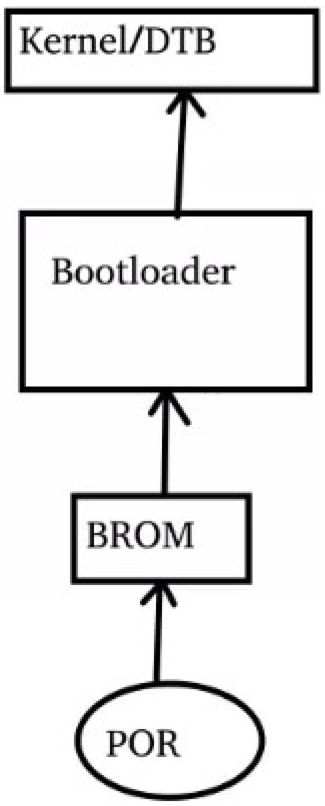
```
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
pthread_t tid1,tid2;
int var=5;
```

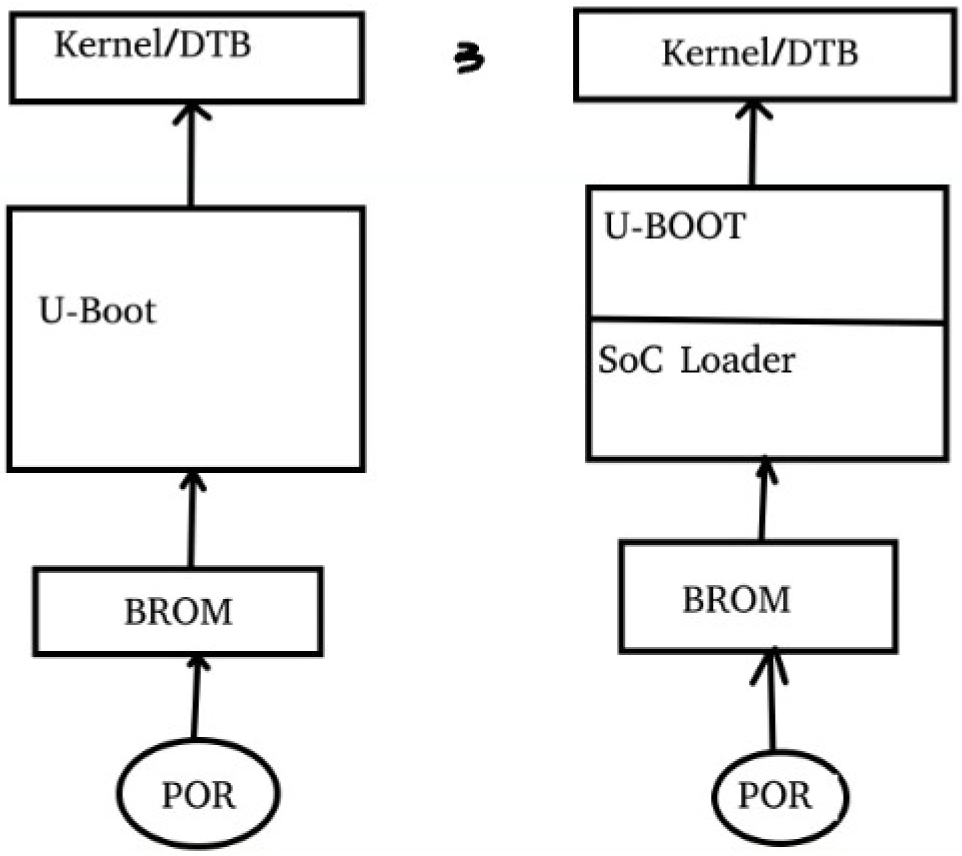
```
void *f1(void *);
void *f2(void *);
int main()
{
        printf("in main\n");
        pthread_create(&tid1,NULL,f1,NULL);
        pthread_create(&tid2,NULL,f2,NULL);
        pthread_exit(NULL);
        return 0;
}
void *f1(void *)
{
        while(1)
        {
                printf("in f1....var=%d\n",var);
                ++var;
                sleep(3);
        }
        printf("thread1 exitng\n");
}
void *f2(void *)
{
        while(1)
        {
                printf("in f2....var=%d\n",var);
                sleep(1);
        }
        printf("thread2 exiting\n");
        pthread_exit(NULL);
}
```

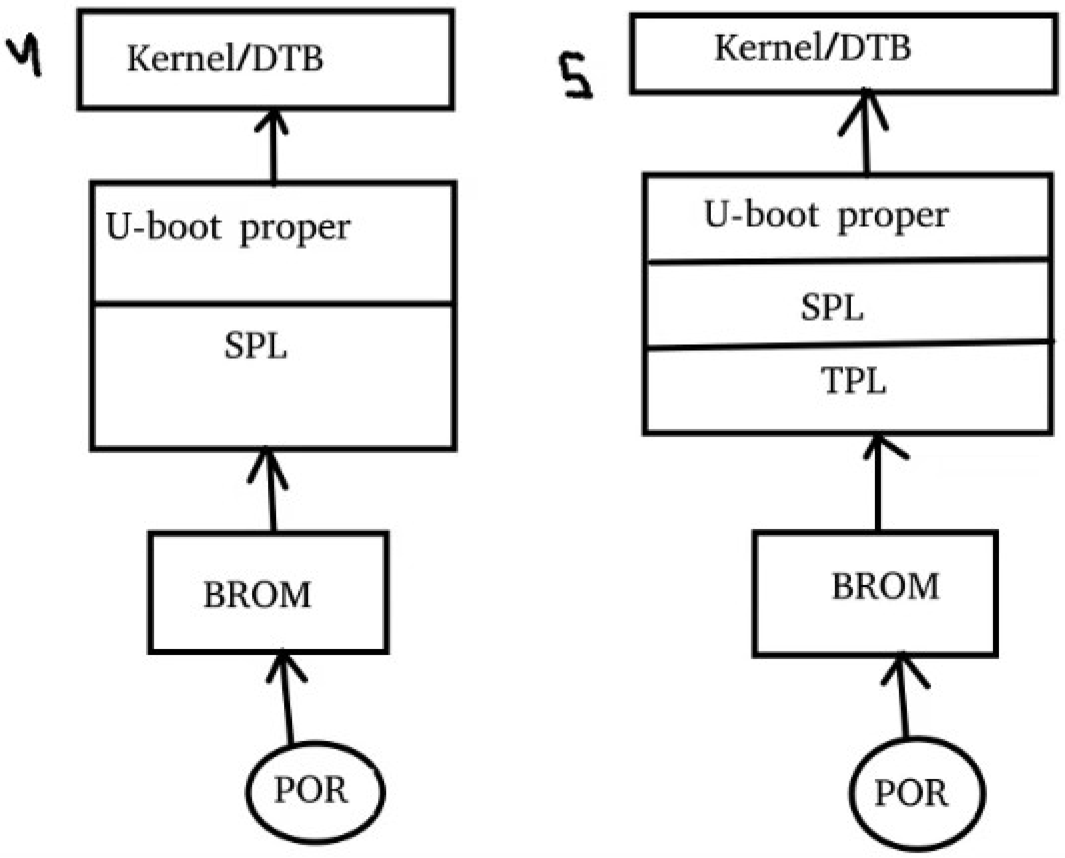
Guide:Sandeep **Embedded Linux** **06/12/23**

U-Boot BootLoader:

- ->Hardware support
- ->Architecture/SOC
- ->X86,MIPS,PowerPc,RISCV
- ->Board : It will support approx 190+ board vendor,1085+ diff board.


Kernel/DTB
Bootloader
BROM
POR


Kernel/DTB
U-Boot
BROM
POR
3
Kernel/DTB
U-BOOT
SoC Loader
BROM
POR



->Hardware support

SPL: Secondary program loader

- Build from the same source as uboot.
- Reduce the size.
- Used to init system and start U-BOOT or linux.

TPL: Tortiary program loader

- Build from the same source as uboot.
- Interms of size it is small than SPL.
- Used on severaly limited system(ONE NAND).

Basics U-boot Commands:

U-boot shell(HUSH)->It is similar like bourne shell.

- **help**(help echo,help bdinfo,help usb)

- **bdinfo:**prints board info structure.
- **Coinfo:**print the console device and information.
- **Echo:**print the text.
  Eg: echo(hello_ptg)->hello_ptg.
- **Bdinfo:** it will give architecture number.
- **Memory access:** mw,md->use for reading/writing to memory and register it support byte(.b)/word(.w)/long(.l)register.

Guide:Ramesh **Process Management** **06/12/23**

Task:

Program on Orphan Process

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
int main()
{
    pid_t p;
    p=fork();
    if(p==0) //child
    {
       printf("I am child having PID %d\n",getpid());
       printf("My parent PID is %d\n",getppid());
    }
    else //parent
    {
       printf("I am parent having PID %d\n",getpid());
       printf("My child PID is %d\n",p);

    }
 }
```

Program on Zombie Process

```
#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

   pid_t child_pid;
```

```
    child_pid = fork();

    if (child_pid == 0) {

        printf("Child process is exiting\n");

        exit(0);

    } else if (child_pid > 0) {

        printf("Parent process sleeping for a while\n");

        sleep(10);


        printf("Parent process exiting\n");

    } else {

        perror("Fork failed");

        return 1;

    }

return 0;
}
```

Guide:Sandeep **Embedded Linux** **07/12/23**

U-Boot Commands:

byte(.b) word(.w) long(.l) with md/mw.

cmp:	memory compare
bdinfo:	print board information
bootm:	boot application image from memory.
clock:	set processor clock.
cp:	memory copy
date:	get/set/reset date and time.
ls:	list of files.
md:	memory display
mm:	memory modification
mtest:	memory write
printenv:	print environment variable
saveenv:	save environment variable
setenv:	set environment variable
usb:	USB subsystem
USBboot:	boot from usb device

```
=>md.l      0x81000000   0x8
=>mw        0x81000000   0x1234abcd
```

81000000:1234abcd 00000000 00000000 00000000 ..4......
support for reading multiple units
at a time(default 0x40)

=>**md.w 0x810000 0x8**
81000000 :abcd 1234 0000 0000 0000 0000 0000 0000 0000 0000 ..4.....

=>**md.b :0x81000000 0x8**
8100000000 : cd ab 34 12 00 00 00 00 00 00 ..4....

=>**I2c Commands:**
• iloop<chip_address> : loops,reading a set of i2c addresses.
Eg: iloop 0x4a
• imd <chip_address> : Display I2c 0x4a
• imm <chip_address> : modify i2c memory
• imw <chip_address> :fils with value an i2c memory

=>**Network Commands:**
• dhcp : request an Ip address from DHCP server
• ping <ping_address> : eg:- ping google.com
• boot[load address][boot filename]: boots the image over network using the boot/tftp protocol.

=>**Simple and recursive variable:**
# setenv mynumber 123456789
#printenv mynumber=>Recursive variable:
#setenv dumpaddr md.b \${addr}\${bytes}
#printenv dumpaddr
dumpaddr md.b \${addr}\${bytes}
#setenv addr 2c000
#setenv bytes 5
#run dumpaddr
0002c000 : 00 00 00 00 0000.....

=>**Conditional expression:**
• if true ; then echo “hello”; else echo “byteptg”;fi
hello
• false || echo “false!”
false

=>**Test command:**
• env set i 6
• test $i -lt 5
• echo $?

Guide:Ramesh **Process Management** **07/12/23**

Task on Thread:

```
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
pthread_t tid1,tid2;
int var=5;
void *f1(void *);
void *f2(void *);
int main()
	{
	printf("in main\n");
	pthread_create(&tid1,NULL,f1,NULL);
	pthread_create(&tid2,NULL,f2,NULL);
	pthread_exit(NULL);
	return 0;
}
void *f1(void *)
{
	while(1)
	{
		printf("in f1....var=%d\n",var);
		++var;
		sleep(3);
	}
	printf("thread1 exitng\n");
}
void *f2(void *)
{
	while(1)
	{
		printf("in f2....var=%d\n",var);
		sleep(1);
	}
	printf("thread2 exiting\n");
	pthread_exit(NULL);
}
```

Guide:Ramesh **Thread and PThread** **08/12/23**

PCB (Process Control Block):

A Process Control Block in OS (PCB) is a data structure used by the operating system to manage information about a process.It is also known as Task Control Block (TCB) in some operating systems. It contains information about the process state, memory allocation, CPU usage, I/O devices, and other resources used by the process.

- The main purpose of a PCB is to enable the OS to manage multiple processes efficiently by keeping track of the state of each process and allocating system resources accordingly.
- When a process is created, the OS creates a PCB for that process and stores all the necessary information about the process in it.
- The OS then uses the information in the PCB to manage the process and ensure that it runs efficiently.
- The information stored in a PCB is used by the OS to make decisions about which process to run next, which resources to allocate, and when to switch between processes.

Task:

Write a C program that uses two threads to calculate the sum of elements in two halves of an array concurrently. The program should include a function calculateSum that takes the array, start index, end index, and a reference to a result variable as parameters. The main function should create an array of integers, spawn two threads to calculate the sum of each half of the array, and then wait for both threads to finish. Finally, calculate the total sum and print the result.

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
struct mydata
{
        int *arr,start,end,*result;
};
void *calculatesum(void *arg)
{
         struct mydata* data=(struct mydata*)arg;
         int minsum=0;
         for(int i=data->start;i<=data->end;i++)
{
         minsum+=data->arr[i];
        }
        *(data->result)=minsum;
        pthread_exit(NULL);
}
int main()
{
         int arr[100],n,i;
         printf("enter no of elements");
        scanf("%d",&n);
         for(i=0;i<n;i++)
 {
         scanf("%d",&arr[i]);
         }
         int r1,r2;
        pthread_t t1,t2;
```

```
            struct mydata data1={arr,0,n/2-1,&r1};
            struct mydata data2={arr,n/2,n-1,&r2};
            pthread_create(&t1,NULL,calculatesum,(void*)&data1);
            pthread_create(&t2,NULL,calculatesum,(void*)&data2);
           pthread_join(t1,NULL);
           pthread_join(t2,NULL);
           int totalsum=r1+r2;
           printf("total sum:%d\n",totalsum);
           return 0;
}
```

Guide:Ramesh **Pthread** **11/12/2023**

Task 1

1.
```
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
pthread_t tid1,tid2;
int var=5;
void *f1(void);
void *f2(void);
int main()
        {       printf("in main\n");
        pthread_create(&tid1,NULL,f1,NULL);
        pthread_create(&tid2,NULL,f2,NULL);
        pthread_join(tid1,NULL);
        pthread_join(tid2,NULL);
        pthread_exit(NULL);
        return 0;
}
void *f1(void )
{
        static int v1=6;
                printf("in f1....var=%d\n",var);
                ++var;
                sleep(3);
        printf("thread1 exitng\n");
        pthread_exit(&v1);
}
void *f2(void)
{
        static int v2=8;

                printf("in f2....var=%d\n",var);
                sleep(1);

        printf("thread2 exiting\n");
        pthread_exit(&v2);
}
```

Task 2

```
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
pthread_t tid1,tid2;
int var=5;
void *f1(void *);
void *f2(void *);
int main()
        {
        printf("in main\n");
        pthread_create(&tid1,NULL,f1,NULL);
        pthread_create(&tid2,NULL,f2,NULL);
        pthread_exit(NULL);
        return 0;
}
void *f1(void *)
{
        for(int i=1;i<5;i++)
        {
                printf("in f1....var=%d\n",var);
                ++var;
             //   sleep(3);
        }
        sleep(3);
        printf("thread1 exitng\n");
}
void *f2(void *)
{
        for(int j=1;j<5;j++)
        {
                printf("in f2....var=%d\n",var);
                //sleep(1);
        }
        printf("thread2 exiting\n");
        pthread_exit(NULL);
}
```

- YP Board Explanation
- Build and compiled steps
- Tought about Flashing tool
- Explained about different Display’s

Guide:Ramesh **Pthread** **12/12/23**

Task:

/*1. Write a simple C program using pthreads to create two threads that print "Hello from Thread 1" and "Hello from Thread 2."*/

```
#include<stdio.h>
#include<pthread.h>
pthread_t tid1,tid2;
void *f1(void *);
int main()
{
	pthread_create(&tid1,NULL,f1,"Hello from Thread 1");
	pthread_create(&tid2,NULL,f1,"Hello from Thread 2");
	pthread_join(tid1,NULL);
	pthread_join(tid2,NULL);
	pthread_exit(NULL);
	return 0;
}
void *f1(void *str)
{
	char *s;
	s=(char*)str;
	printf("%s\n",s);
}
```

/*2.Use condition variables to implement a program with three threads - A, B, and C. Thread A prints "A," B prints "B," and C prints "C." The output should be in the sequence ABCABCABC.*/

```
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
pthread_t tid1,tid2,tid3;
void *f1(void*);
int main()
{
	pthread_create(&tid1,NULL,f1,"A");
	pthread_create(&tid2,NULL,f1,"B");
	pthread_create(&tid3,NULL,f1,"C");
	pthread_join(tid1,NULL);
	pthread_join(tid2,NULL);
	pthread_join(tid3,NULL);
	pthread_exit(NULL);
}
void *f1(void *c)
{
	char *ch;
```

```
        ch=(char*)c;
        for(int i=0;i<3;i++)
        {
        printf("%s",ch);
        sleep(2);
        }
}
```

Guide:Sandeep **Embedded Linux** **12/12/23**

scrcpy : virtual screen copy
adb : debugging purpose
communicate host to target

ADB(Android Debug Bridge):

command line tools to communicate with device

- $ adb wait-for-device
- $ adb devices
- $ adb shell
- $ adb reboot
- $ adb reboot bootloader
- $ adb push <File.txt/.c>>/path>
- $ adb pull

Guide:Ramesh **Process Management** **19/12/23**

Mutex and Semaphore:

In the Operating System, Mutexes and Semaphores are kernel resources that provide synchronization services .Synchronization is required when multiple processes are executing concurrently, to avoid conflicts between processes using shared resources.

Mutex:

Mutex is a specific kind of binary semaphore that is used to provide a locking mechanism. It stands for Mutual Exclusion Object. Mutex is mainly used to provide mutual exclusion to a specific portion of the code so that the process can execute and work with a particular section of the code at a particular time.

Mutex uses a priority inheritance mechanism to avoid priority inversion issues. The priority inheritance mechanism keeps higher-priority processes in the blocked state for the minimum possible time. However, this cannot avoid the priority inversion problem, but it can reduce its effect up to an extent.

Semaphore:

A semaphore is a non-negative integer variable that is shared between various threads. Semaphore works upon signaling mechanism, in this a thread can be signaled by another thread. Semaphore uses two atomic operations for process synchronisation:

- Wait (P)
- Signal (V)

The **wait** operation decrements the value of the semaphore, and the **signal** operation increments the value of the semaphore. When the value of the semaphore is zero, any process that performs a wait operation will be blocked until another process performs a signal operation.

Semaphores are of two types:

1. **Binary Semaphore :**
   This is also known as a mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.

2. **Counting Semaphore :**
   Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

GUIDE:Sandeep **Kernel Module Programming**

Task:

```
1. #include<linux/module.h>
#include<linux/printk.h>

int init_module(void)
{
        pr_info("hello");
        return 0;
}
void cleanup_module(void)
{
        pr_info("Goodbye");
}
MODULE_LICENSE("GPL");


2.#include<linux/module.h>
#include<linux/kernel.h>
#include<linux/init.h>

static int hello_data __initdata=3;

static int __init hello_init(void)
{
      printk(KERN_INFO "hello %d\n",hello_data);
      return 0;
```

```
}
static void __exit hello_exit(void)
{
        printk(KERN_INFO "goodbye\n");
}

module_init(hello_init);
module_exit(hello_exit);
MODULE_LICENSE("GPL");


3.#include<linux/init.h>
#include<linux/kernel.h>
#include<linux/module.h>
#include<linux/moduleparam.h>
#include<linux/printk.h>
#include<linux/stat.h>

MODULE_LICENSE("GPL");

static short int myshot=1;
static int myint=123;
static long int mylong=9999;
static char *mystring="blah";
static int myintarray[2]={12,45};
static int arr_argc=0;

module_param(myshot,short,S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP);
module_param(myint,int,S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH);
module_param(mylong,long,S_IRUSR);
module_param(mystring,charp,0000);
module_param_array(myintarray,int,&arr_argc,0000);

static int __init hello_init(void)
{
        int i;
        printk(KERN_INFO "hello,world\n============ \n");
        printk(KERN_INFO "myshort is a short integer:%hd\n",myshot);
        printk(KERN_INFO "myint is an integer:%d\n",myint);
        printk(KERN_INFO "mylong is a long integer:%ld\n",mylong);
        printk(KERN_INFO "mystring is a string:%s\n",mystring);

        for(i=0;i<(sizeof myintarray/sizeof(int));i++)
        {
                printk(KERN_INFO "myintarray[%d]=%d\n",i,myintarray[i]);
        }
        printk(KERN_INFO "got %d argument for myintarray.\n",arr_argc);
        return 0;
}
static void __exit hello_exit(void)
{
        printk(KERN_INFO "Goodbye,PTG");
```

```
}
module_init(hello_init);
module_exit(hello_exit);


4.#include<linux/kernel.h>
#include<linux/init.h>
#include<linux/module.h>

MODULE_LICENSE("GPL");
static int a[7];
static int count=0;

module_param_array(a,int,&count,0000);

static int peak_init(void)
{
        int i=0;
         if(a[0]>a[i+1])
                printk(KERN_INFO "%d\n",a[i]);
        for(i=1;i<count-1;i++)
        {
                if((a[i]>a[i+1]) &&(a[i]>a[i-1]))
                        printk(KERN_INFO "%d\n",a[i]);}
                 if(a[i]>a[i-1])
                        printk(KERN_INFO "%d\n",a[i]);
      return 0;
}
static void peak_exit(void)
{
        printk(KERN_INFO "bye\n");
}

module_init(peak_init);
module_exit(peak_exit);


5.#include<linux/module.h>
#include<linux/kernel.h>
#include<linux/init.h>

static int min,max;
MODULE_LICENSE("GPL");

module_param(min,int,S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP);
module_param(max,int,S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP);

static int prime_init(void)
{
            int i,cnt;
            for(min;min<=max;min++)
            {
```

```
                     cnt=0;
                    for(i=1;i<=min;i++)
                    {
                            if(min%i==0)
                                    cnt++;
                    }
                    if(cnt==2)
                            printk(KERN_INFO "%d ",min);
            }
            printk(KERN_INFO "\n");
            return 0;
}
static void prime_exit(void)
{
            printk(KERN_INFO "OK SHAMEEM GARU\n");
}

module_init(prime_init);
module_exit(prime_exit);
```

```
6.#include<linux/kernel.h>
#include<linux/module.h>
#include<linux/init.h>
#include<linux/kdev_t.h>
#include<linux/fs.h>

dev_t dev=0;


static int hello_init(void)
{
            if((alloc_chrdev_region(&dev,0,1,"drive"))<0)
            {
                    pr_err("cannot allocted");
                    return -1;
            }
            pr_info("module inserted successfully\n");
            return 0;
}

static void hello_exit(void)
{
            unregister_chrdev_region(dev,1);
            pr_info("module removed successfully\n");
}

module_init(hello_init);
module_exit(hello_exit);


MODULE_LICENSE("GPL");
```

7.
```
#include<linux/kernel.h>
#include<linux/module.h>
#include<linux/init.h>
#include<linux/kdev_t.h>
#include<linux/fs.h>
#include<linux/err.h>
#include<linux/device.h>

dev_t dev=0;
static struct class *dev_class;

static int hello_init(void)
{
	if((alloc_chrdev_region(&dev,0,1,"Driver"))<0)
	{
		pr_err("module created successfully\n");
		return -1;
	}
	pr_info("minor=%d , major=%d",MINOR(dev),MAJOR(dev));

	dev_class= class_create("chr_dr2");
	if(IS_ERR(dev_class))
	{
		pr_err("cannot created class\n");
		goto r_class;
	}

	if(IS_ERR(device_create(dev_class,NULL,dev,NULL,"Driver")))
	{
		pr_err("cannot created device\n");
		goto r_device;
	}
	pr_info("module inserted successfully\n");
	return 0;
r_device:
	class_destroy(dev_class);
r_class:
	unregister_chrdev_region(dev,1);
	return -1;
}

static void hello_exit(void)
{
	device_destroy(dev_class,dev);
	class_destroy(dev_class);
	unregister_chrdev_region(dev,1);
	pr_info("module removed successfully\n");
}

module_init(hello_init);
```

```
module_exit(hello_exit);

MODULE_LICENSE("GPL");
```