OS Concepts & Sys Programs:-

* Scheduling
* Memory Management
* File System
+ Earlier topics - Processes, Threads, IPC, Signals

Dev Tools - gcc/g++ options, GNU Tools,
Makefiles, static & dynamic libs

Kernel Development - Linus Torvalds
GNU Software (Free software)
- Richard M Stallman
GNU Linux

Embedded Linux:-

OS Architecture:-
* Kernel (core/sub systems)
* Drivers
* Libraries
* Utils (shell, dev tools,
monitoring)

Distributions:-
* Ubuntu, Debian
* Fedora, RHEL
* Backtrac
* Kali
* CentOS

and many more

Linux Kernel
---
std C library
/usr/lib
libc.a or libc.so
libm.a, libm.so

CPU Families:-

ARM:-
Arduino Atmega328x
STM32F4 Cortex-M4F
MSP432
Tiva Series
and many more
ARM Cortex-M series

High end Targets:-

RaspberryPi BCM28xx
BeagleBoneBlack AM335x

iMX series targets - 6, 7,8 (sabre, nitrogen, wand)
Snapdragon series targets (Dragonboard)
Some TI Boards

Target Board - Single Board Computer (SBC)
Evaluation Board /Development kit
System On Chip(SOC) - CPU, SRAM, Flash, I/O pins on same chip

CPU Family : Cortex-A series (A8, A9, A53)
Multicore
Co-Processor , e.g. DSP, GPU, FPU

Activity:- Survey of
SBCs/DKs/EKs
SOC Behind
CPU Family, how many cores, instruction set family,e.g. armv7-a
Any co-processor
What are other peripherals

Yocto

Qemu - cross platform emulator

Target:- Virtual Express A9, ARM Cortex-A9 CPU

What is required to emulate:-
* kernel image - core + static modules
* device tree blob (dtb)
* root file system image (rootfs)
    - libs, additional drivers(dynamic modules),utils
* bootloader

To generate above, what we need
* kernel source - kernel image + dtb file
* pre-built rootfs (Can generate your own at later stage)
* Cross Tool Chain, to generate code for architecture
* Qemu for emulation / Real Target Board

Choose a clean workspace: ~/eworkdir

```
arm-linux-gnueabi-gcc hello.c –o h1.out
arm-linux-gnueabi-gcc hello.c –o h2.out –static

gcc test.c -c
gcc sum.c -c
gcc sqr.c -c
nm test.o
nm sum.o
nm sqr.o
objdump -t test.o
objdump -t sum.o sqr.o
gcc test.o sum.o sqr.o -o sall.out -static
ls -lh sall.out
strip sall.out            # removes symbol table
ls -lh sall.out

arm-linux-gnueabi-gcc test.c -c
arm-linux-gnueabi-gcc sum.c -c
arm-linux-gnueabi-gcc sqr.c -c
arm-linux-gnueabi-nm test.o
arm-linux-gnueabi-nm sum.o
arm-linux-gnueabi-nm sqr.o
arm-linux-gnueabi-objdump -t test.o
arm-linux-gnueabi-objdump -t sum.o sqr.o
```

```
gcc hello.c -o n.out
arm-linux-gneuabi-gcc hello.c -o c.out
file n.out c.out

readelf -h n.out
readelf -h c.out
```

What is ELF? Executable and Linkable format
object files, libraries, executables

```
arm-linux-gnueabi-readelf -a d1.out | grep .so
arm-linux-gnueabi-objdump -x d1.out | grep NEEDED


# TODO:- try ldd on natively built, dynamically linked executables
```

```
command > file          stdin
command >> file         stdout
command < file          stderr
command 2> file
command | tee file
command &> file
```

```
int a=10;
int b;
static float x=2.3f;
static float y;
const int cval=100;
const int dval;
volatile const int vc;

void f1() {  }
static void f2() {  }

int main() {
   int p,q,r;
}
```

```
gcc demo.c -c
nm demo.o
objdump -t demo.o
size demo.o

arm-linux-gnueabi-gcc demo.c -c
arm-linux-gnueabi-nm demo.o
arm-linux-gnueabi-objdump -t demo.c
arm-linux-gnueabi-size demo.o
```

D, d	.data
C, b	COM, .bss
R, r	.rodata
T, t	.text
U	undefined

d,b,r,t	-- internal linkage only (static prefix)
D, R, T	-- external linkage (no static prefix)

```
arm-linux-gnueabi-ar rc sum.o sqr.o –o libsample.a
```

who calls main? main returns to whom?

startup code , cleanup code part of std C library

glibc is the impl, in the name libc.a or libc.so

find /usr/lib -name 'libc.*'
find /usr/lib -name 'libm.*'
find /usr/lib -name 'libpthread.*'
find /usr/lib -name 'libstdc++.*'
whereis libc

gcc-linaro-7.5.0-2019.12-x86_64_arm-linux-gnueabi
/ arm-linux-gnueabi/libc/usr/lib
---
if libsample.so is copied to ~/mylibs
==> LD_LIBRARY_PATH=/home/root/mylibs ./d1.out
export LD_LIBRARY_PATH=/home/root/mylibs
./d1.out

Let's assume many so files are /opt/mylibs

std lib locations:-
/lib
/usr/lib
/usr/local/lib

/opt/mylibs
echo "/opt/mylibs" >> /etc/ld.so.conf

* Simple Boot
* Cross compilation
* Bootloaders,U-Boot
* Device Tree
* Custom kernel build