./

Learning Report – Kernel Programming



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
|  |  | Aditya Mewara |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

Table of Contents

[1. WHAT DO YOU MEAN BY KERNEL? 4](#_Toc68326545)

# WHAT DO YOU MEAN BY KERNEL?

* A kernel is the central part of an operating system. It manages the operations of the computer and the hardware, most notably memory and CPU time
* It decides which process should be allocated to processor to execute and which process should be kept in main memory to execute.

**There are three types of kernels:**

* 1. **A monolithic kernel**
* It is one of types of kernel where all operating system services operate in kernel space. It has dependencies between systems components. It has huge lines of code which is complex
* Advantage

It has good performance.

* Disadvantage

It has dependencies between system component and lines of code in millions.

* 1. **A micro kernel**
* It is kernel types which has minimalist approach. It has virtual memory and thread scheduling. It is more stable with less services in kernel space. It puts rest in user space.
* Advantage

It is more stable.

* Disadvantage

There are lots of system calls and context switches.

* 1. **Hybrid Kernel**
* It is the combination of both monolithic kernel and microkernel. It has speed and design of monolithic kernel and modularity and stability of microkernel.
* Advantage

It combines both monolithic kernel and microkernel.

* Disadvantage

It is still similar to monolithic kernel.

# WHAT DO YOU MEAN BY MODULES?

* **Modules are** pieces of code that **can** be loaded and unloaded into the **kernel** upon demand.
* They extend the functionality of the kernel without the need to reboot the system.
* The kernel consists of a set of kernel modules that interact with each other, each performing a specific function. Some kernel modules perform software functions exclusively, while others (such as device drivers) control the operation of system hardware components.

1. **Activity QEMU installation**

* QEMU is a generic and open source machine emulator and virtualizer.
* QEMU is used to emulate devices and certain privileged instructions and requires either the KQEMU or KVM kernel modules and the host operating system

**Installing QEMU on ARM based architecture**

* + sudo apt install qemu-system-arm

**Running QEMU by ZImage and vexpress dtb file**

* qemu-system-arm -M vexpress-a9 -m 1024 -serial stdio \ -kernel zImage -dtb vexpress-v2p-ca9.dtb \ -sd rootfs.img -append "console=ttyAMA0 root=/dev/mmcblk0 rw"

1. **Activity TOOLCHAIN Installation**

Installing soft load on ARM Architecture

sudo apt install gcc-arm-linux-gnueabi

1. **Download Kernel Source**

Downloading from linux tar.xz from the source and extract it in a new folder

And then

Obtain the zImage and vexpress dtb file

Linux Commands:

make ARCH=arm mrproper

make ARCH=arm vexpress\_defconfig

1. **Building Kernel Modules:**
   1. **Simple Hello Module:**

* Step 1 : Building the hello.c file and writing the contents
* Step 2: make file and writing the contents ( obj-m += hello.o )
* Cross compile using make
  + make –C ${KSRC} M=${PWD} modules ARCH=arm, CROSS\_COMPILE=arm-linux-gnueabi-
* testing on target
  + sudo mount –o loop,rw,sync rootfs.img /mnt/rootfs
  + sudo cp hello.ko /mnt/rootfs/home/root
  + sudo umount /mnt/rootfs
  1. **Simple hello Module with init and exit function**
* Building the hello.c file and writing the contents
* make file and writing the contents
  + obj-m += hello.o
  + KSRC = (where you have linux tar.xz location)
  + all: make –C ${KSRC} M=${PWD} modules
  + clean: make –C ${KSRC} M=${PWD} clean
* Cross compile using make command
* Testing on the target
  1. **Hello module with parameters**
* Building the hello.c file and writing the contents
* **The contents added to be are:**
* int ndevices=1
* module\_param(ndevices,int,S\_IRUGO);
* make file and writing the contents
* make file and writing the contents
* Now in host i.e QEMU pass the arguments like insmod ndevices = 5 or by default it will be 1
  1. **Module Dependency simple**
* Building the hello.c file and writing the contents
  + - The contents added to be are :
    - The functions and variable are present in the hello.c file
    - EXPORT\_SYMBOL\_GPL(xvar);
    - EXPORT\_SYMBOL\_GPL(sayHello);
    - make file and writing the contents
      * **obj-m += simple.o  
        all:  
         make -C /home/user/eworkspace/kernel\_ws/ksrc M=${PWD} modules ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi-  
        clean:  
         make -C /home/user/eworkspace/kernel\_ws/ksrc M=${PWD} modules ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi-**
      * Now open the emulation using tempboot location
      * run command to print the contents insmod (.ko) file
      * dmesg will display the contents of the file
  1. **Module Dependency sample**
* Building the hello.c file and writing the contents
  + - The contents are added to be are apart from simple
    - **extern int xvar;**
    - **extern void sayHello(void);**
    - Then after importing the module from simple we can use the functions defined in the simple module by printing in the sample module
    - We need to first run the simple module and then sample module so that we can use the functions present in the simple module