# Operating System

## QEMU

$sudo apt install libglib2.0-dev libgcrypt20-dev zlib1g-dev gcc-multilib autoconf automake bison flex

Ssudo apt install libpixman-1-dev libz-dev libtool libtool-bin libsdl1.2-dev

$tar xvf qemu-3.0.0.tar.xz

$cd qemu-3.0.0/

$sudo ./configure --disable-kvm --target-list="i386-softmmu x86\_64-softmmu"

$sudo make -j6

$sudo make install

//以上是QEMU的安装

$mkdir ~/6.828

$cd ~/6.828/

$git clone https://pdos.csail.mit.edu/6.828/2017/jos.git lab

$cd lab/

$sudo make

$sudo make qemu

## PC Architecture

The purpose of computing is to gain insight

### PC with people insight

|  |
| --- |
| Problem |
| Algorithm |
| Program/Language |
| Runtime System (VM, OS, MM) |
| ISA (Instruction Set Architecture): Interface/contract between Software and Hardware. |
| Microarchitecture: An implementation of the ISA |
| Logic |
| Devices |
| Elections |

## OS Basic

Kernel headers are all in /usr/src/linux-headers-4.4.0-116/include/linux //worthy thoroughly learning

Makefile

obj-m = hello.o

KVERSION = $(shell uname -r)

all:

make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules

clean:

make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean

Kernel Operation:

$ tail -f /var/log/kern.log

$ insmod hello.ko

$ rmmod hello

### Definition

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs.

### OS Structure

1. the services that the system provides
2. the interface that it makes available to users and programmers
3. the components and their interconnections

### A view of operating system services

### System Call

In computing, a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on.

System calls provide an essential interface between a process and the operating system.

#### Types of System Calls

1. Processs control

* End, abort
* Load, execute
* Create process, terminate process
* Get process attributes, set process attributes
* Wait fot time
* Wait event, signal event
* Allocate and free memory

1. File manipulation

* Create file, delete file
* Open, close
* Read, write, reposition
* Get file attributes, set file attributes

1. Device manipulation

* Request device, release device
* Read, write, reposition
* Get device attributes, set device attributes
* Logically attach or detach devices

1. Information maintenance

* Get time or date, set time or date
* Get system data, set system data
* Get process, file, or device attributes
* Set process, file or device attributes

1. Communications

* Create, delete communication connection
* Send, receive messages
* Transfer status information
* Attach or detach remote devices

## Process Management

### 1 Process

* A program itself is not a process; a process is an active program and related resources.
* A program becomes a process when an executable file is loaded into memory.

### 2 Process classification

Processes can be classified as either I/O-bound or processor-bound.

The scheduling policy in a system must attempt to satisfy two conflicting goals: fast

process response time (low latency) and maximal system utilization (high throughput).

### 3 Process scheduling

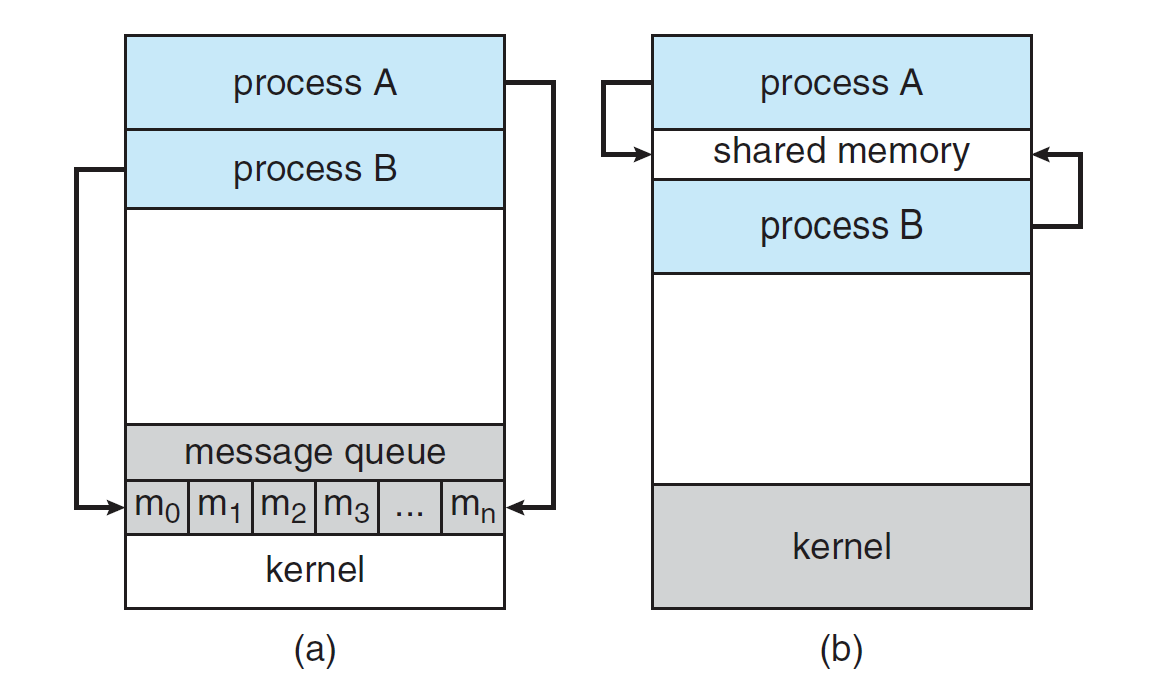
#### Scheduling queues

1. Job queue: consists of all processed in the system
2. Ready queue: consists of the processes that are residing in main memory and are ready and waiting to execute
3. Device queue: consists of processes waiting for a particular I/O devuce

### 3 Interprocess Communication

There are two fundamental models of IPC

1. Shared memory
2. Message passing
3. RPC (RPCs are a form of IPC)



### 4 Thread

#### Signal handle and asynchoronous procedure call

A signal is used in Unix systems to notify a process that a particular event has occurred.

#### Thread pool

The general idea behind a thread pool is to create a number of threads at process startup and place them into a “pool”, where they sit and wait for work.

#### Thread creation

What resources are used when a thread is created? How do they differ from those used when a process is created?

Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation. Creating a process requires allocating a process control block (PCB), a rather large data structure. The PCB includes a memory map, list of open files, and environment variables. Allocating and managing the memory map is typically the most time-consuming activity. Creating either a user or kernel thread involves allocating a small data structure to hold a register set, stack, and priority.

## Memory Management

### 1 address binding

### 2.dynamic loading and dynamic linking

# OS Service

## Program Execution

### compilation system

* Preprocessing phase(cpp): include header file.
* Compilation phase(cc1): translate modified source program (. i file) to assembly program (. s file).
* Assenmbly phase(as): translate text file to binary file.
* Linking phase(ld): translate relocatable object to executable object.

## Filesystem

## Resource Allocation

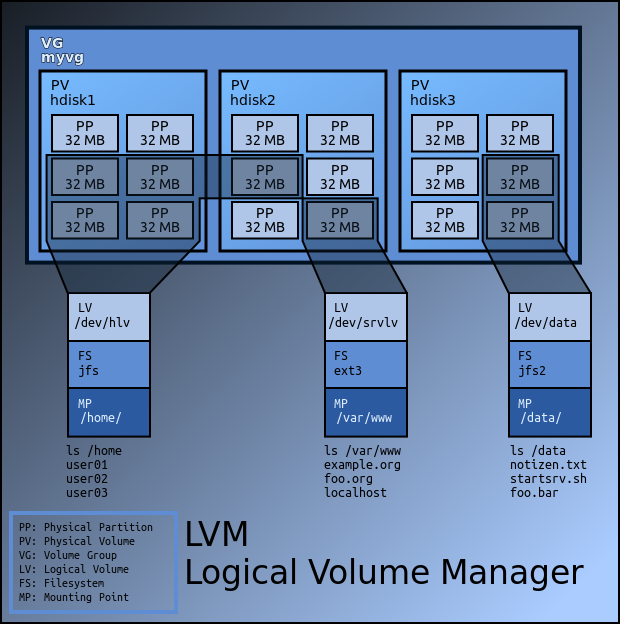
## I/O Operation

# Linux

## LVM

Ref: https://www.howtoforge.com/linux\_lvm

In Linux, Logical Volume Manager (LVM) is a device mapper target that provides logical volume management for the Linux kernel.



$ fdisk /dev/sdb # add a new physical partition, Changed system type of partition 1 to 8e

(Linux LVM)

$ pvcreate /dev/sdb1 #create physical volume, use command pvdisplay

$ vgcreate <vg\_name, eg fileserver> /dev/sdb1 <pv1, pv2, …>#create volume group, use command vgdisplay

$ lvcreate --name share --size 40G fileserver #create logical volume from VG fileserver,

Use command lvdisplay

$ lvrename fileserver media films

$ lvremove /dev/fileserver/films

$ lvextend -L1.5G /dev/fileserver/media

$ lvreduce -L1G /dev/fileserver/media

$ mkfs.ext3 /dev/fileserver/share and mount

Now, you can use df -h to check out the situation, finally you can modify the file /etc/fstab to mount the filesystem automatically when system boots up

## resize a logical volume

$ mkfs.xfs -f /dev/sdb

$ pvcreate /dev/sdb

$ vgextend centos /dev/sdb #volume group entend

$ lvextend -L +25G /dev/centos/root # logical volume extend

$ xfs\_growfs /dev/centos/root # Finally perform an resize to resize the logical volume