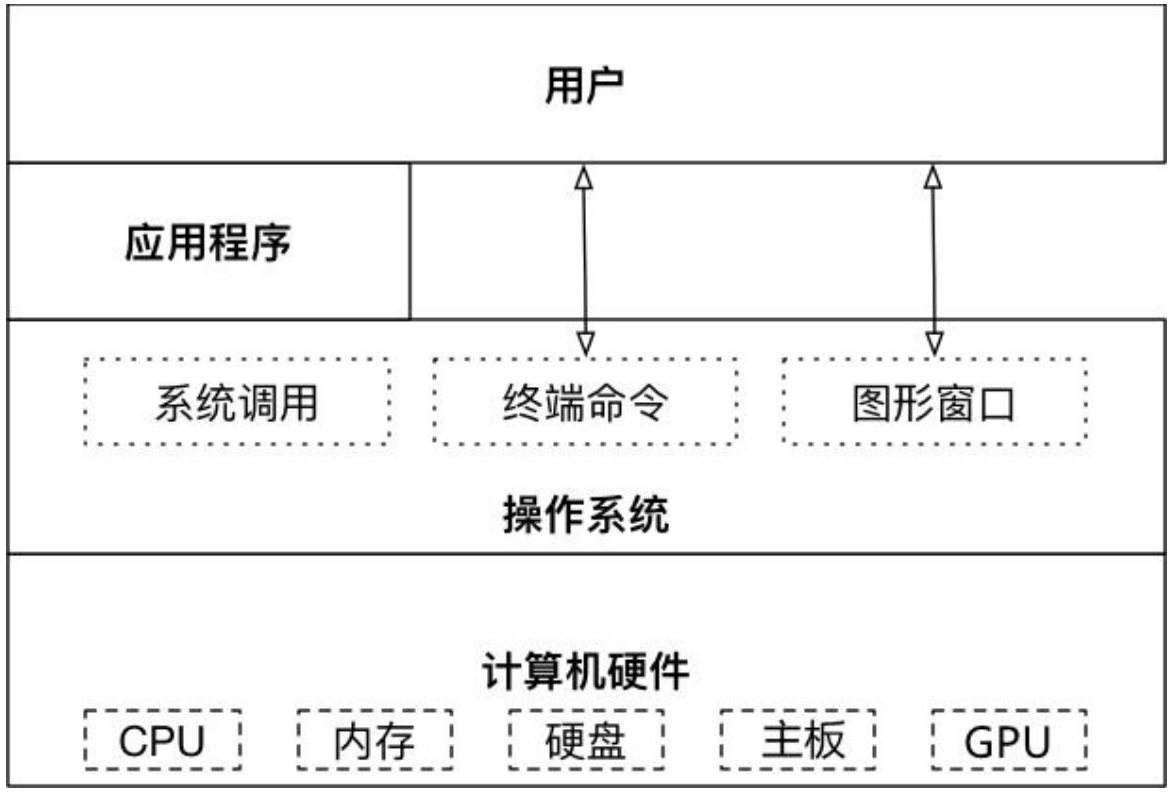
CUDA ON ARM 夏令营 – L4T UBUNTU 基础

NVIDIA企业级开发者社区 李奕澎

课程目录

- 操作系统
- Linux操作系统发展历史
- L4T Ubuntu与X86 Ubuntu系统的区别
- 实验平台与基本开发环境
- Ubuntu文件管理–目录结构
- Ubuntu用户权限管理
- Ubuntu常用命令的基本使用
- Ubuntu网络管理与SSH
- Makefile的基本介绍与编写规则

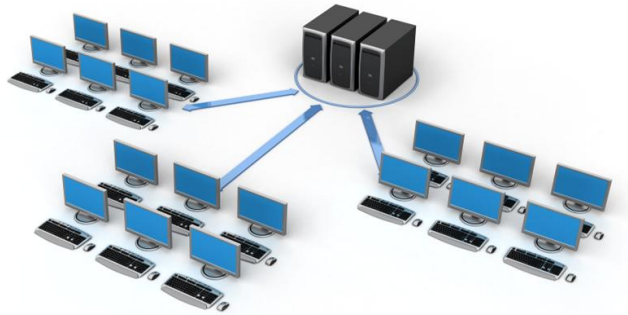
操作系统



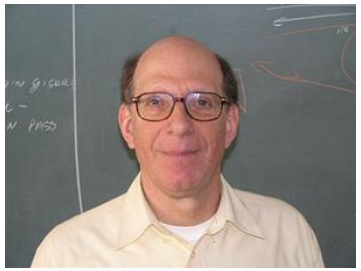
操作系统的作用：

- 是现代计算机系统中最基本和最重要的系统软件。
- 是配置在计算机硬件上的第一层软件，是对硬件系统的首次扩展。
- 主要作用是管理好硬件设备，并为用户和应用程序提供一个简单的接口，以便于使用。
- 而其他的诸如编译程序、数据库管理系统，以及大量的应用软件，都直接依赖于操作系统的支持。

LINUX操作系统发展历史



- Linux发展简史
  - 1965
    - 贝尔实验室和麻省理工合作
    - 多使用者、多任务、多层次的操作系统
    - 同时支持300台终端
  - 1969
    - Ken Tompson
      - 星际旅行游戏的源动力
      - Unix操作系统原型
  - 1973
    - Ken Tompson & Dennis Ritchie
      - 重写Unix合力完成UNIX操作系统
  - 1987
    - Andrew S. Tanenbaum
      - MINIX
  - 1991
    - Linus
      - Linux
        - Ubuntu
        - Centos
        - Redhat
        - Fedora



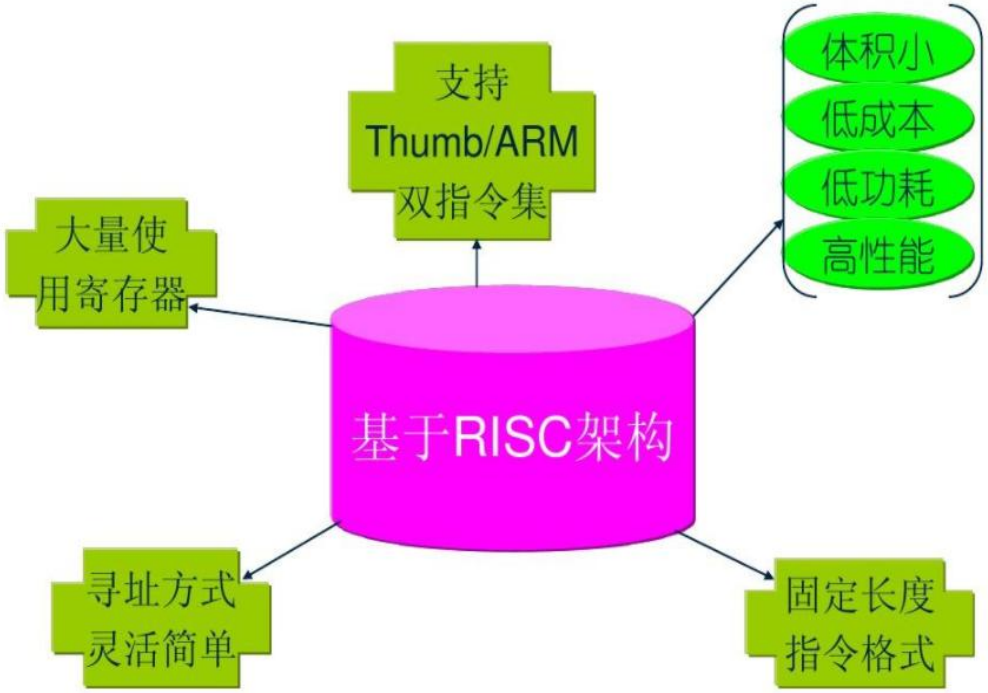
X86 UBUNTU与L4T UBUNTU的区别

X86 Ubuntu： 是指运行在X86架构CPU的linux ubuntu版本的操作系统。

L4T Ubuntu： L4T 是linux for tegra的缩写，Tegra是集成了ARM架构的CPU和NVIDIA的GPU的处理器芯片，所以L4T Ubuntu就是为运行在基于arm架构的Tegra芯片上的linux ubuntu版本的操作系统，它是专门为Tegra定制的Ubuntu特殊版本。

X86 Ubuntu与L4T Ubuntu主要区别：

- 应用场景：ARM处理器定位于嵌入式平台，应用在开发板、边缘设备、智能设备上；X86定位于桌面PC和服务器。
- ARM是为了低功耗高效率设计的，而X86是为了追求高性能。
- 设计架构：ARM是精简指令集（RISC）架构；x86是复杂指令集(CISC)架构。
- ARM几乎都采用Linux的操作系统；X86多为window系统也可采用linux操作系统。



实验平台及开发环境

	JETSON NANO
GPU	128 Core Maxwell 0.472 TFLOPs (FP16)
CPU	4 core ARM A57 @ 1.43 GHz
Memory	4 GB 64 bit LPDDR4 25.6 GB/s
Storage	16 GB eMMC
Video Encode	4K @ 30 | 4x 1080p @ 30 | 8x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 16x 720p @ 30 | (H.264/H.265)
Camera	12 (3x4 or 4x2) MIPI CSI-2 DPHY 1.1 lanes (1.5 Gbps)
WiFi/BT	Requires external chip
Display	HDMI 2.0 or DP1.2 | eDP 1.4 | DSI (1 x2) 2 simultaneous
UPHY	1 x1/2/4 PCIE 1 USB 3.0
SATA	None
Other I/Os	1xSDIO / 2xSPI / 3xI2C / UART / I2S / GPIOs
USB OTG	Not supported
Mechanical	69.6mm x 45mm 260 pin edge connector, No TTP

GPU	NVIDIA Volta 架構搭配 384 個 NVIDIA CUDA® 核心及 48 個 Tensor 核心
CPU	6 核心 NVIDIA Carmel ARM® v8.2 64 位元 CPU 6 MB L2 + 4 MB L3
DL 加速器	2x NVDLA 引擎
視覺加速器	7 向 VLIW 視覺處理器
記憶體	8 GB 128 位元 LPDDR4x 51.2GB/s
儲存裝置	microSD（不含卡片）
視訊編碼	2x 4Kp30 | 6x 1080p60 | 14x 1080p30 (H.265/H.264)
視訊解碼	2x 4Kp60 | 4x 4Kp30 | 12x 1080p60 | 32x 1080p30 (H.265) 2x 4Kp30 | 6x 1080p60 | 16x 1080p30 (H.264)
相機	2x MIPI CSI-2 D-PHY 通道
連接功能	Gigabit 乙太網路、M.2 鍵位 E（包含 WiFi/BT）、M.2 鍵位 M（NVMe）
顯示器	HDMI 和 DP
USB	4x USB 3.1、USB 2.0 Micro-B
其他	GPIO、I^2C、I^2S、SPI、UART
機體	103 mm x 90.5 mm x 31 mm

https://www.nvidia.cn/autonomous-machines/embedded-systems/jetson-nano/product-development/

Jupyter Lab基本介绍

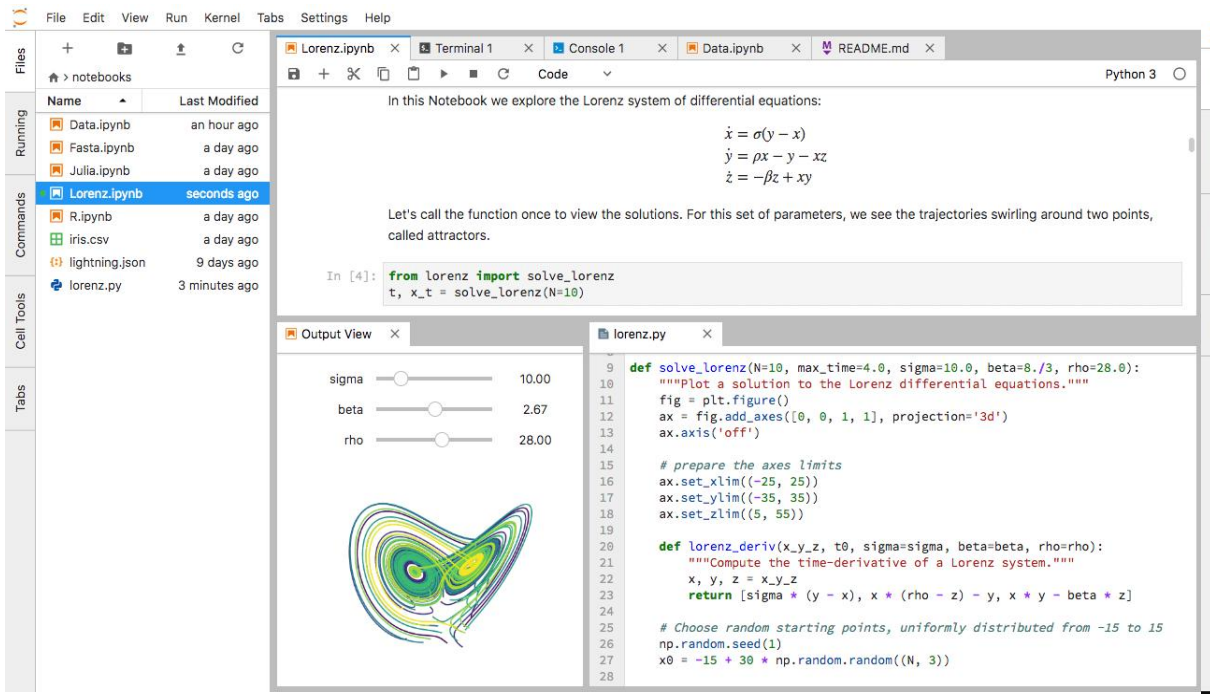
JupyterLab：数据科学生产工具，它作为一种基于web的集成开发环境，你可以使用它编写代码的notebook、操作终端、编辑markdown文本、打开交互模式、查看csv文件及图片等功能。

安装方法：

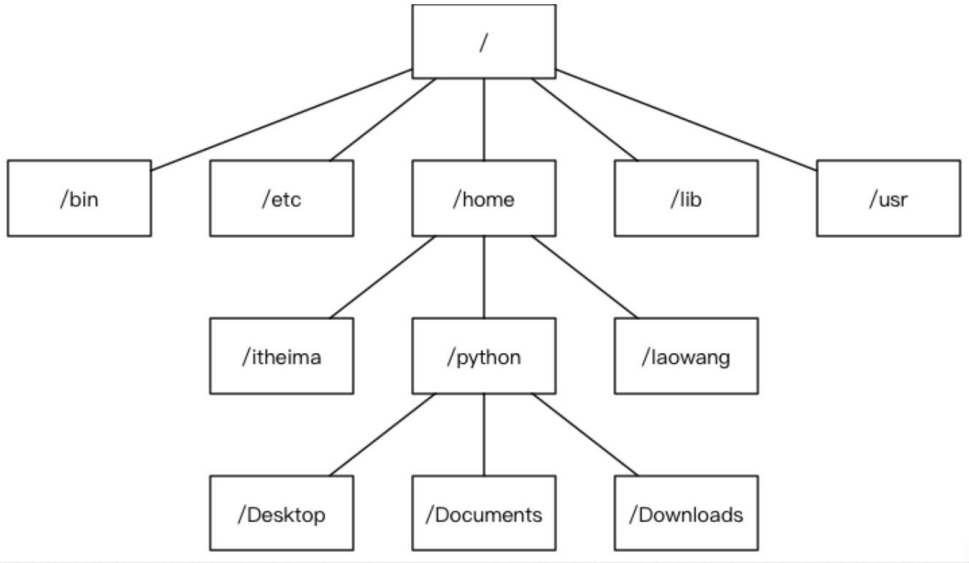
```
#pip
pip install jupyterlab

#conda
conda install -c conda-forge jupyterlab
```

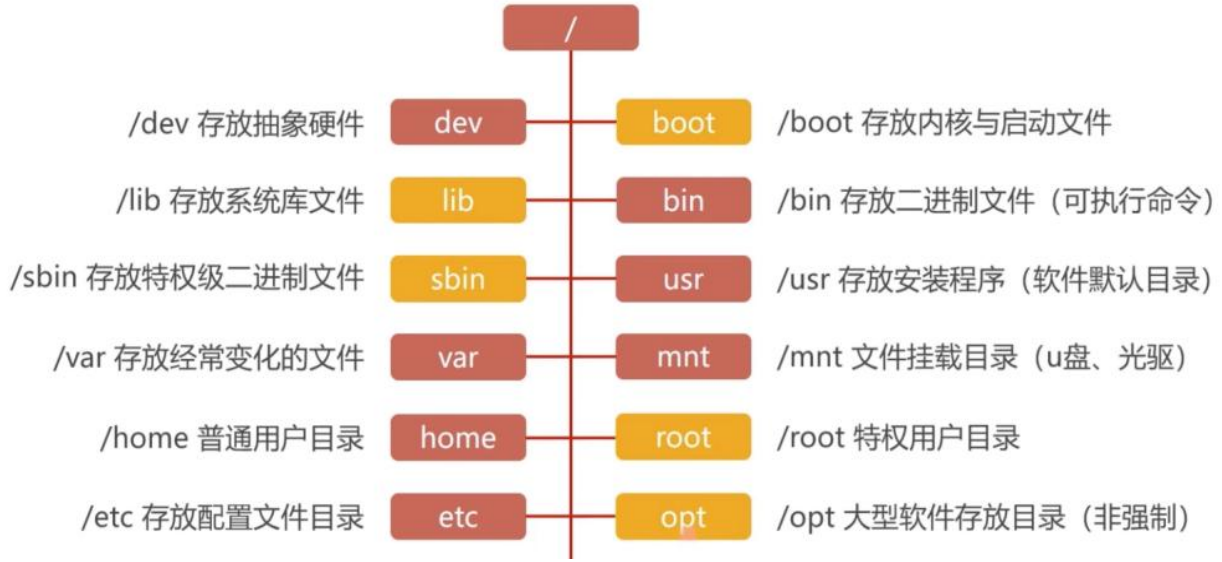
启动方法：

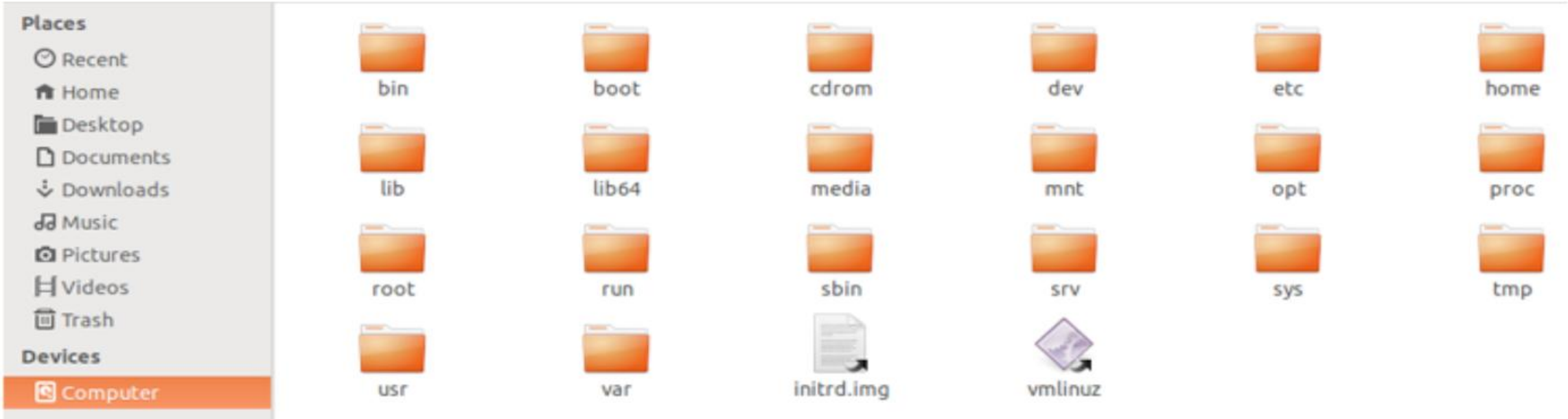
```
jupyter lab
```



UBUNTU的文件管理 - 目录结构







UBUNTU的权限管理

超级用户：

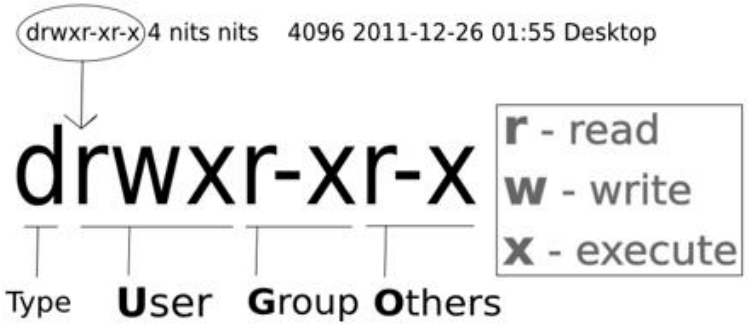
```
$ su      $ su -
```

exit	退出当前用户
sudo + 命令	使得用户可在自己的环境下,执行需要 root 权限的命令

用户组:

```
nvidia@nvdc-48:~$ ls -l
total 3692
drwxrwxrwx 10 nvidia nvidia    4096 7月  11 14:24 CUDA_on_ARM
-rw-rw-r--  1 nvidia nvidia 3068452 7月   4 12:28 CUDA_on_ARM.zip
-rw-rw-r--  1 nvidia nvidia  703509 7月   5 12:43 makeCourse.zip
```

- （User）文件所有者。
- （Group）群组 ，我们可以在Ubuntu系统中为团　队建立一个群组。
- （Others）其他人，不属于文件所有者或文件所属群组成员的用户便都是其他人。



UBUNTU的权限修改

修改形式字母法：chmod （u g o a） （+ - =） （r w x） （文件名）

[u g o a]	含义
u	user 表示该文件的所有者
g	group 表示与该文件的所有者属于同一组(group)者，即用户组
o	other 表示其它用户组
a	all 表示这三者皆是

[+ - =]	含义
+	增加权限
-	撤销权限
=	设定权限

r	读	设置为可读权限
w	写	设置为可写权限
x	执行权限	设置为可执行权限

```
chmod ugo+r file1.txt

chmod ug+w,o-w file1.txt file2.txt
```

修改形式数字法：

用法：chmod + 数字组合 + 文件名

数字组合一般包含三个数字：

第一个数字对应字母法的用户u（user）　r (read) ----------> 4

第二个数字对应字母法的用户g（group）　w (write) ----------> 2

第三个数字对应字母法的用户o（other）　x (excute) ----------> 1

举例说明：

数字法：chmod 777 文件名 <----------对应----------> 字母法： chmod u+rwx, g+rwx, o+rwx 文件名

第一个数字7：代表用户 u 的权限 rwx， 4 (r) + 2 (w) + 1 (x) = 7

第二个数字7：代表用户 g 的权限 rwx， 4 (r) + 2 (w) + 1 (x) = 7

第三个数字7：代表用户 o 的权限 rwx， 4 (r) + 2 (w) + 1 (x) = 7

对应关系：

```
数字法：chmod 755 filename 对应 字母法： chmod u+rwx, g+rx, o+rx filename
数字法：chmod 751 filename 对应 字母法： chmod u+rwx, g+rx, o+x  filename
数字法：chmod 765 filename 对应 字母法： chmod u+rwx, g+rw, o+rx filename
```

UBUNTU基本操作命令

查看文件目录:

命令	说明	example
`cd [dir]`	切换到指定目录	
`ls [-options] [dir,file]`	查看指定目录所有内容	cd /home
`pwd [-options]`	打印当前工作目录	
`cat [-options] [file]`	查看文件内容/创建文件/文件合并/追加文件内容等	cat 1.txt
`more [-options] file`	分页显示文件内容	

文件新建/删除:

命令	说明	
`touch [-options] file`	若文件不存在则创建文件,否则修改文件最后编辑时间	touch 1.txt
`mkdir [-options] dir`	创建目录。 `-p` 可以**层级创建目录**，如 `mkdir -p a/b/c`	mkdir testdir
`rm [-options] file`	删除文件或目录,删除后**不可恢复**	rm 1.txt
`rmdir [-options] dir`	删除目录,目录必须为空	

UBUNTU基本操作命令

文件拷贝/移动：

命令	说明	example
`cp [-options] source_file target_file`	复制文件或目录	cp 1.txt cp -r dir1/
`mv [-options] source_file/dir target_file/dir`	移动/重命名 文件或目录	mv dir1 dir2

文件搜索：

命令	说明	
`find [path] [-options] [expression]`	在目录中搜索文件	find / -name 1.txt
`grep [-opinions] [pattern] [file]`	在文本文件中查找内容	grep aaa 1.txt

查看时间与日期：

```
$ date      #查看系统时间
$ cal -y    # 查看当年日历
```

UBUNTU基本操作命令

查看空间：

命令	作用
`df [-options] [FILE]`	显示磁盘容量、已用空间、剩余空间、使用例等
`du [-options] [FILE]`	显示目录下文件大小。

```
$ df -h          # 显示磁盘使用情况
```

```
$ du -h Desktop # 显示桌面目录内容尺寸
```

查看与处理进程：

命令	作用
`ps [options]`	查看进程信息。
`top [-options]`	动态显示进程有序详细信息。类似Windows进程排序列表
`kill [-signal] pid`	终止进程

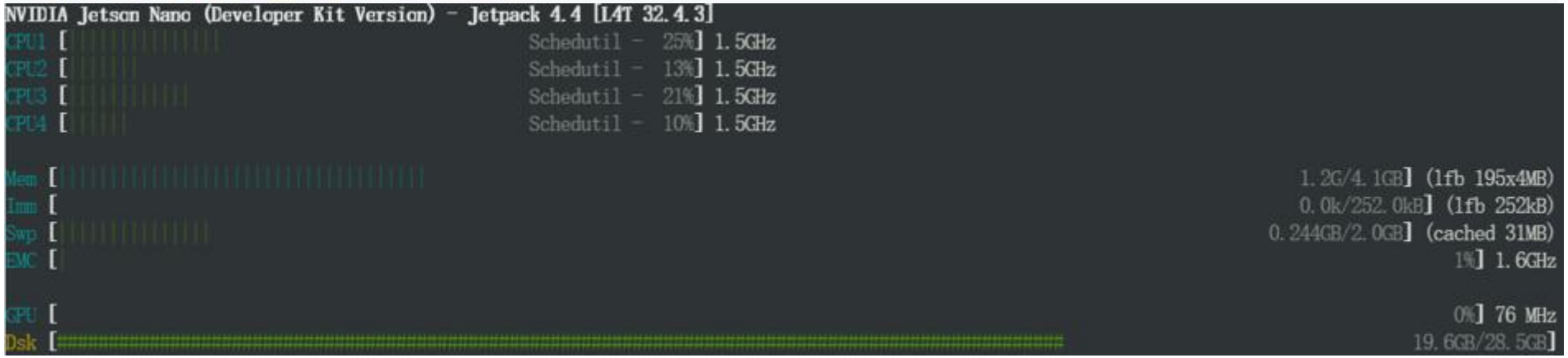
```
# 显示当前用户终端启动进程信息
$ ps
```

```
# 显示所有用户终端启动进程详细信息
$ ps au
```

```
$ kill -9 1278
```

Jtop查看工具：

```
sudo -H pip install -U jetson-stats
```



UBUNTU网络管理

网络管理：

`ifconfig [-options]` 或 `ip addr`	查看或配置网卡信息 `$ ifconfig | grep inet`
`ping [-options] destination`	检测到目标地址连接通讯是否正常

SSH远程连接：

SSH是专为远程登录会话和其他网络服务提供安全性的协议，有效防止远程管理过程中的信息泄漏。

- 传输数据加密，能够防止DNS和IP欺骗。
- 传输数据压缩，加快传输速度。
- Linux中默认已安装SSH客户端，可直接在终端中使用SSH命令。

`ssh [-options] [user@hostname]`	远程连接服务器

```
# 以colin用户登录192.168.1.196的到ssh服务器
$ ssh colin@192.168.1.196
# 以colin用户登录到192.168.1.198的ssh服务器，使用2222端口
$ ssh -p 2222 colin@192.168.1.198
```

Makefile基本介绍与编写规则

什么是Makefile： 当一个工程中的源文件不计数，其按类型、功能、模块分别放在若干个目录中，makefile定义了一系列的规则，可以来指定哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译。makefile就像一个Shell脚本一样，可以实现“自动化编译”，一旦写好，只需要一个make命令，整个工程完全自动编译，极大的提高了软件开发的效率。

Makefile的编写规则：

目标文件： 依赖文件
TAB键 命令

makefile

```
test: test.c
    gcc test.c -o test

```

test.c

```
#include <stdio.h>

int main() {
    printf("hello world\n");
    return 0;
}
```

```
nvidia@nvdc-99:~/CUDA_冬令营/makefiletutorial/example01$ make
gcc test.c -o test
nvidia@nvdc-99:~/CUDA_冬令营/makefiletutorial/example01$ ls
makefile  test  test.c
nvidia@nvdc-99:~/CUDA_冬令营/makefiletutorial/example01$ ./test
hello world
```

多个C的工程文件：

main.c
makefile
max.c
max.h
min.c
min.h

makefile

```
CC = gcc
main: main.c max.o min.o
	$(CC) main.c max.o min.o -o main

max.o: max.c
	$(CC) -c max.c

min.o: min.c
	$(CC) -c  min.c

clean:
	rm *.o main
```

main.c

```
#include "max.h"
#include "min.h"
#include<stdio.h>

int main() {
    int arr[]={1,8,6,3,4};
    int min = find_min(arr,5);
    int max = find_max(arr,5);
    printf("min = %d\n",min);
    printf("max = %d\n",max);
    return 0;
}
```

```
nvidia@nvdc-99:~/CUDA_冬令营/makefiletutorial/example02$ make
gcc -c max.c
gcc -c  min.c
gcc main.c max.o min.o -o main
nvidia@nvdc-99:~/CUDA_冬令营/makefiletutorial/example02$ ls
main  main.c  makefile  max.c  max.h  max.o  min.c  min.h  min.o
nvidia@nvdc-99:~/CUDA_冬令营/makefiletutorial/example02$ ./main
min = 1
max = 8
```

https://blog.csdn.net/kunhe0512/article/details/128623790


THANK YOU
CUDA ON ARM PLATFORM ---学习
GPU编程，迈上AI科研的制高点，让你的职业生涯快人一步！
NVIDIA