hj复习：

用户登录Linux默认所处的路径，都在自己的家目录下

Cd ~ :回到家目录。

在winidows中登录后有一个桌面，桌面用户进入了自己的家目录，在家目录中找到

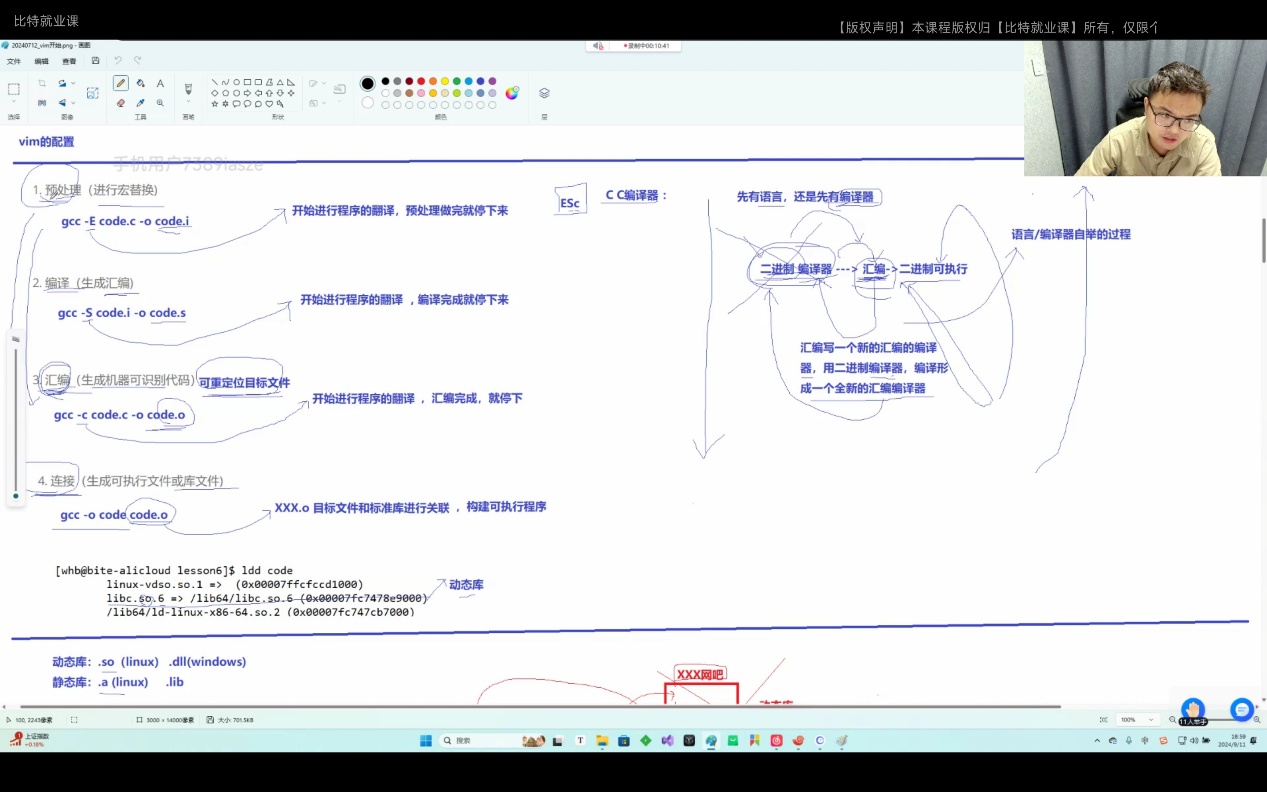
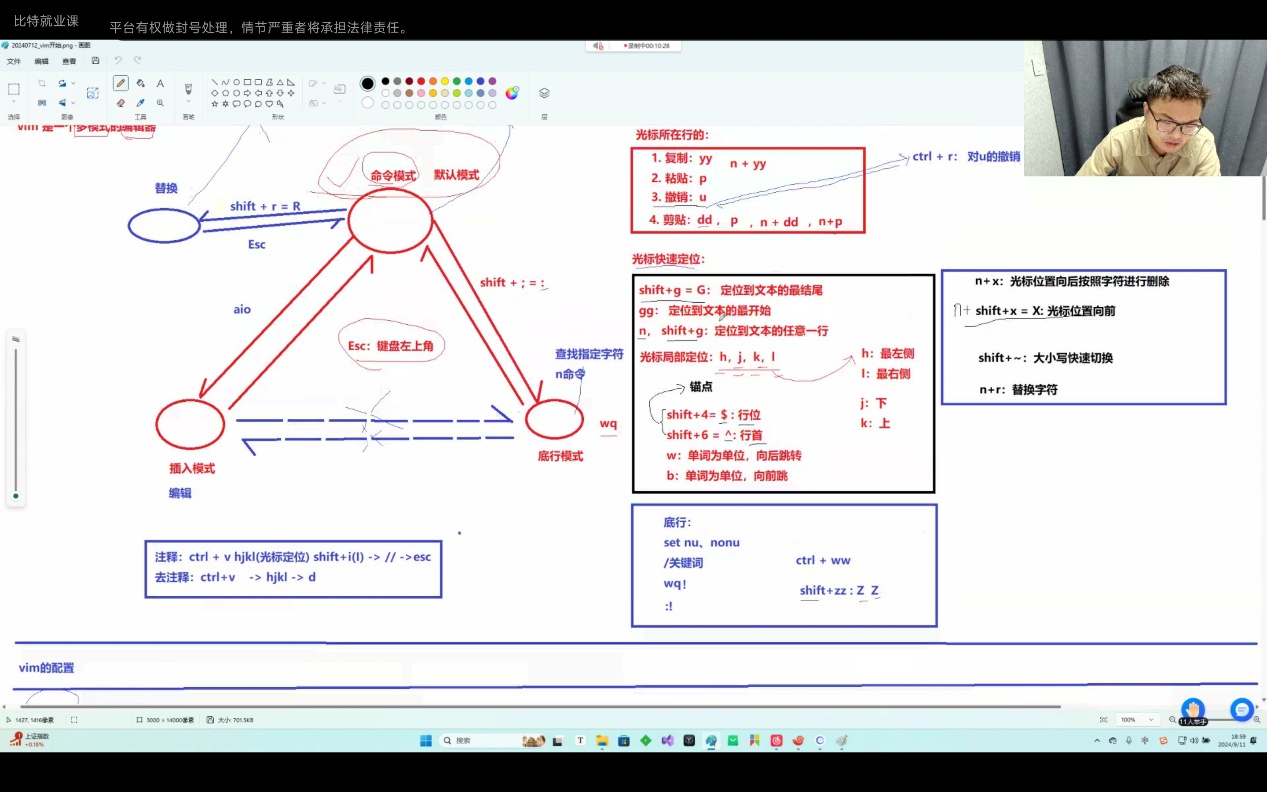
Linux基本结构介绍---》文件命令—》权限

最终权限=起始权限&（~权限掩码）

工具 ----yum---gcc---vim

Linux 软件生态

Vim 命令



云服务器默认没有安装C/C++的静态标准库。

需要用yum安装。

Make+Makefile

Linux调试-gdb/cgdb使用

程序发布有两个版本1debug（添加调试信息） 2release(编译形成可执行程序>取消调试信息)

Gcc/g++编译程序的时候发布默认采用的是release版本

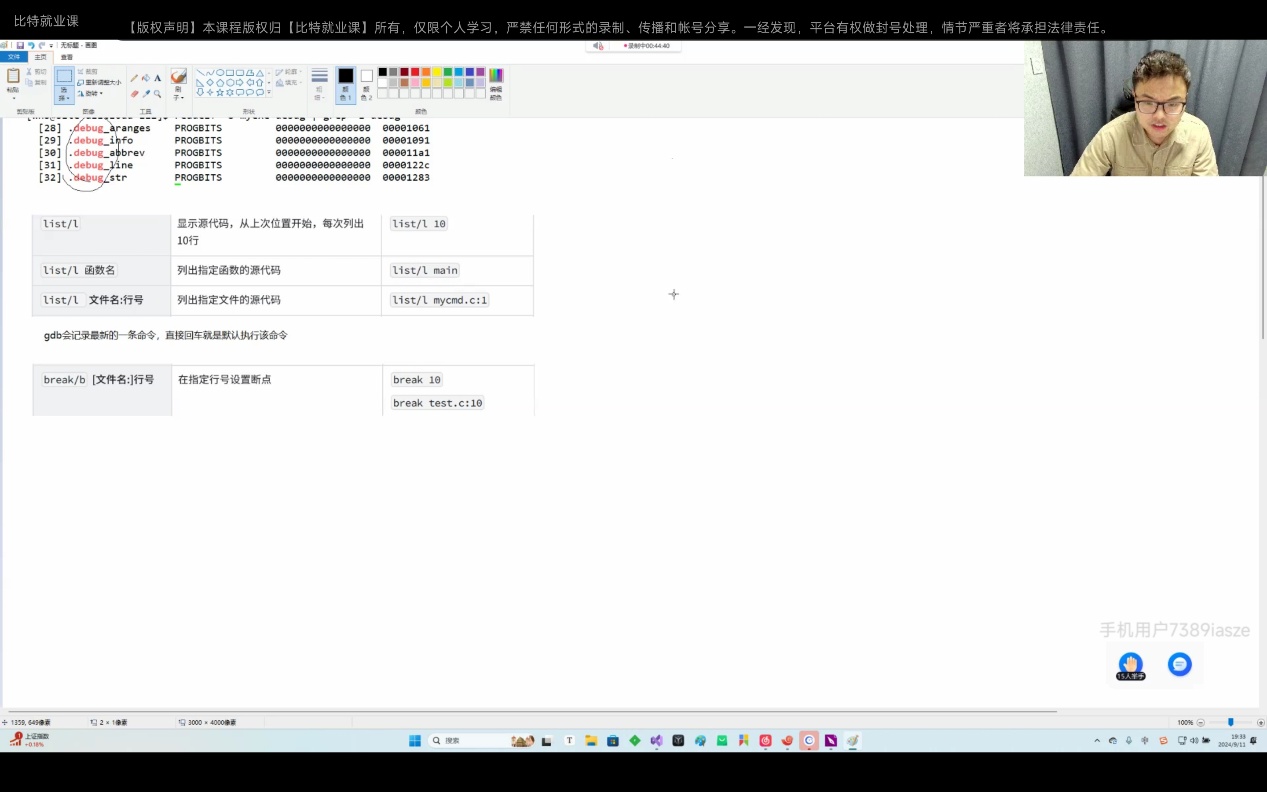
想让发布Debug加参数-g

Readelf –读取一些可执行程序的信息

L 查看源代码

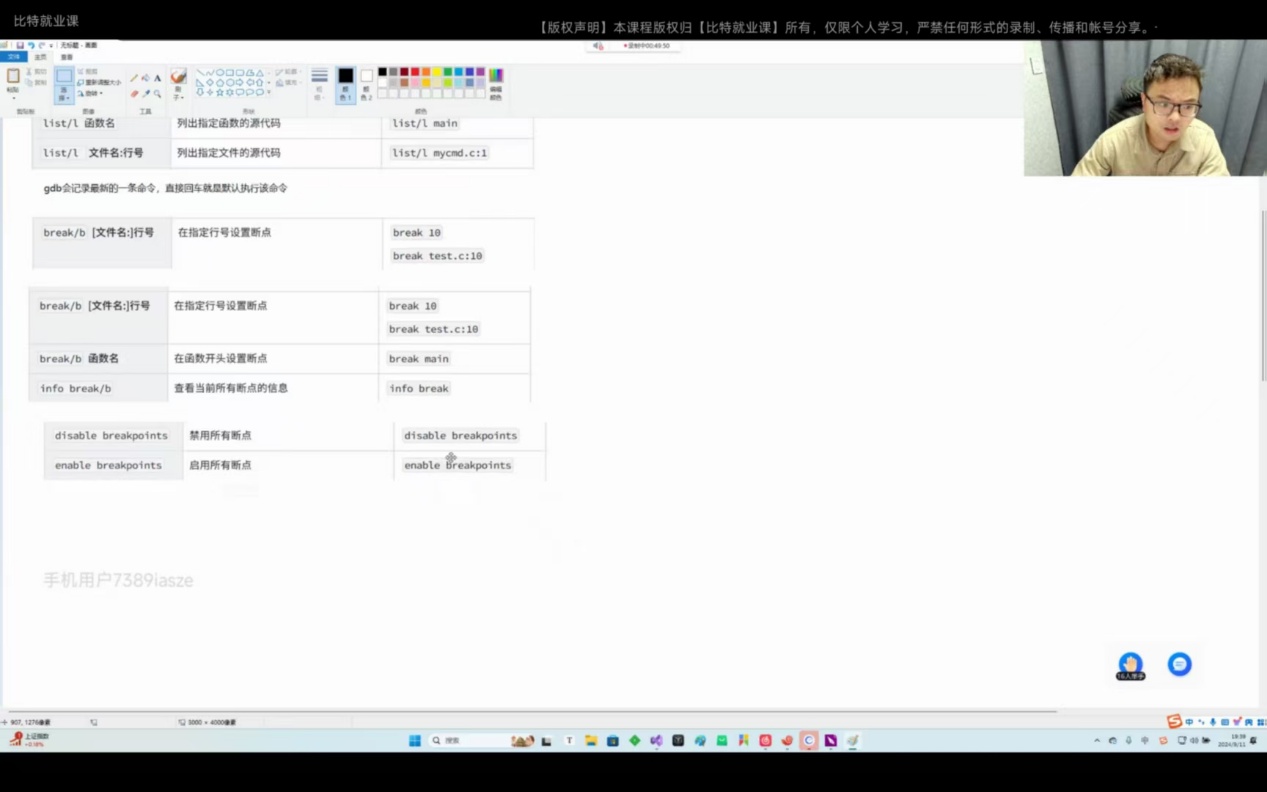
Gdb会记录最新一条的命令直接回车，就是默认执行该命令

B 加行号 ----在指定行号设置断点

D 断点编号 ---删除指定编号断点

B 也可以加指定函数

最常见的就是行号

不想删除断点，而是关掉断点—矢能

R run-----F5

Windows ---F10------linux -----n(next)

F11--------s(step单步执行)

监视窗口---1 display 变量

2 undisplay 常显示编号

P 不想常显示，单纯查看一下

Until 代码行号 直接执行到该行

调试的本质---找到问题

目标解决问题。

调试工具是帮我们找到问题，然后让我们自己解决bug

C (coutinue)---使代码从一个断点直接运行到下一个断点

可以通过打断点限定区域的方式把代码划分为一个个的代码块，用C使 代码快速从一个断点直接运行到下一个断点，快速测试代码块，去缩小问题范围。

以函数为单位确定一个一个的函数有咩有问题

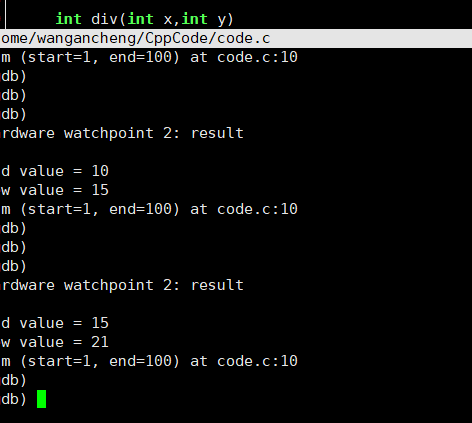
Finish --把当前函数代码跑完

定位问题推荐组合：断点+finish+until+c:对大的代码块进行区间debug

-----快速定位问题的位置

Find locals- --- vs自动窗口---将函数执行过程中临时变量都显示出来

监视一个值---watch---观察重要标志位---值变换 watch才会通知用户



Set var ----对于一个变量的值运行期间就给他改了---作用---发现问题—验证问题—解决问题

条件断点---调试的时候—我们特别关心某个条件---当条件被触发---调试器要告诉我

1. 新增条件断点
2. 给已有条件断点新增 condition 断点编号 条件

---条件满足断点被触动

输入设备：键盘，鼠标，网卡，磁盘，摄像头，话筒。

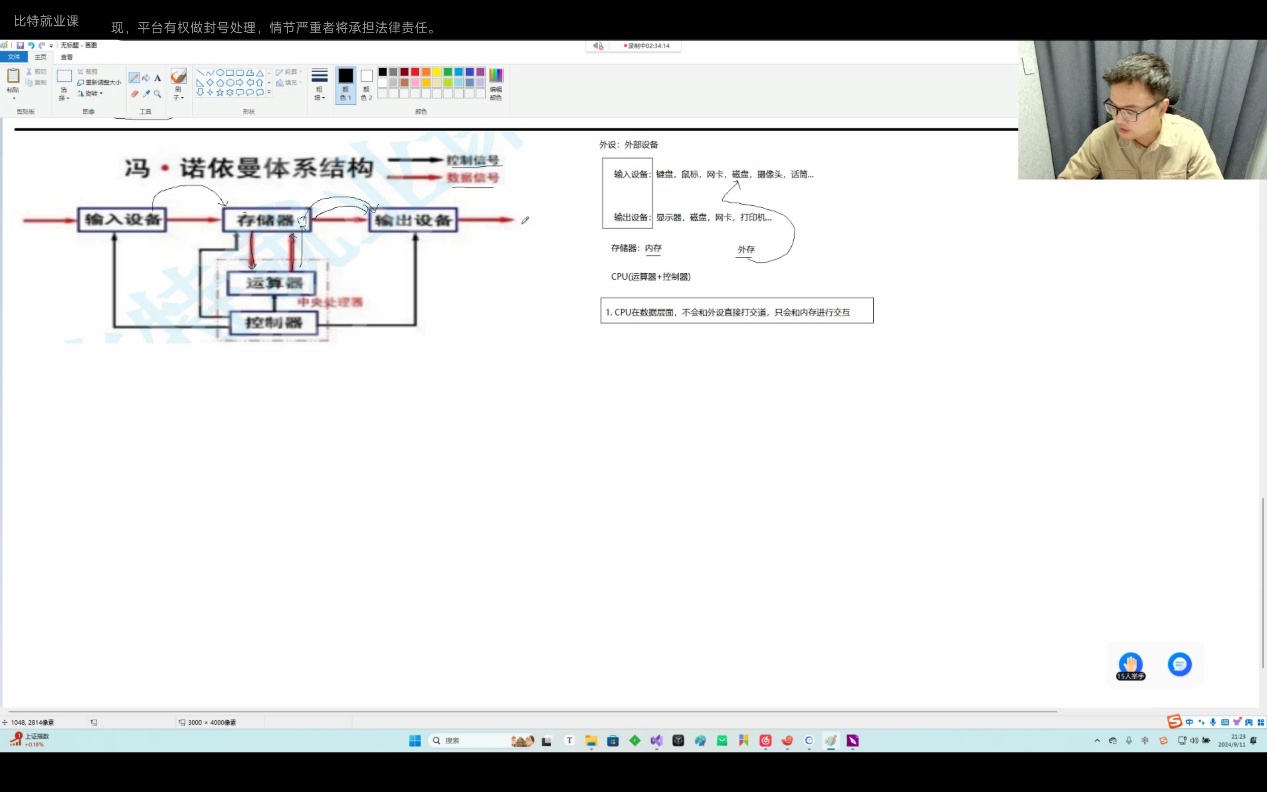
输出设备：显示器，磁盘，网卡，耳机，打印机

二者统一叫做计算机的外设

存储器：内存 外存----磁盘

CPU（运算器+控制器）：

1. CPU 在数据层面上，不回和外设直接打交道，只会和内存进行交互



Cpu处理a=5,b=4;a+b; 不会直接从输入设备读取数据而是先将数据调入存储器（内存）---在从存储器中读取数据-----计算出结果后再将结果传到存储器----从存储器再传入输出设备。------进度条的缓存就是内存

意义：理解一个问题：二进制程序的本质就是文件，文件存储在磁盘上，磁盘是外设

任何程序，运行时，都必须先被（从磁盘）加载到内存

为啥任何程序，运行时，都必须先被（从磁盘）加载到内存？？？

体系结构这么规定  
编译好的二进制程序就是代码加数据---代码需要cpu执行，数据需要cpu 计算

换言之，我们的 代码和数据必须能被CPU访问—但是我们的CPU只会访问内存，不会访问外设----所以我们的CPU要执行程序必须让我们的操作系统将程序从外设调入内存，cpu 从内存中读取。

IO,INPUT.OUTPUT

Input—将数据搬入内存

Output—将数据从存储器刷新到输出设备。

Scanf将数据从键盘读入程序，程序在内存/printf是代码，代码经过编译就会变成可执行程序---》被运行---》就得先被加载到内存----》

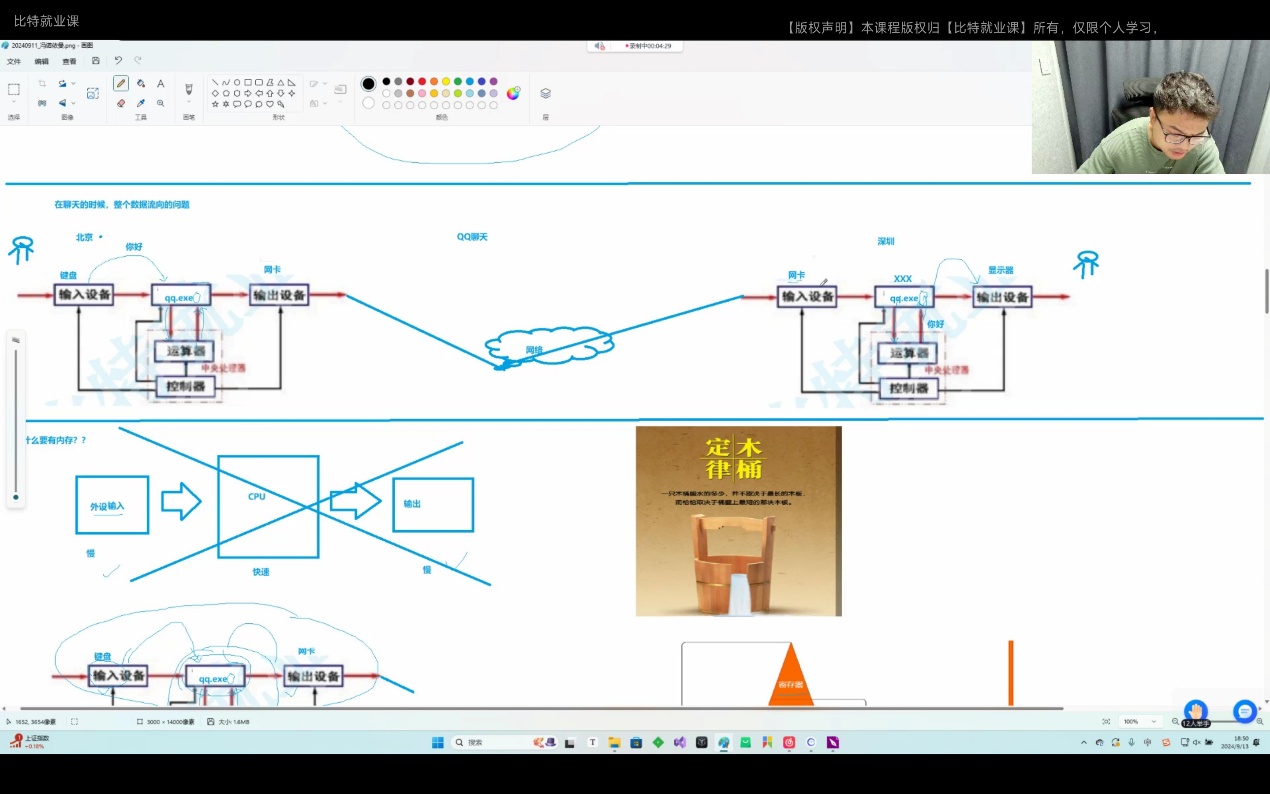
为啥要有内存---

外设很慢毫秒级别cpu很快纳秒级别

无内存根据木桶原理----计算机的效率以外设为主。

在计算机中有个规则越靠近cpu的（寄存器，缓存）越贵，容量小，效率高

远的相反。有了内存，CPU与内存传递数据，外设与cpu传递数据，计算机的效率由内存决定---伟大的发明



数据在计算机内部流转的时候—本质是在不同的数据间拷贝

设备的拷贝效率就是计算机的效率

操作系统内核---kernal

外部程序shell glibc 原生的库，预装软件

操作系统内核工作：进程/任务管理，文件管理，内存管理，驱动管理

操作系统：os本质是已款进行软硬件资源管理的软件

电脑（豪宅）（豪宅管家）---用户（豪宅主人）

为啥要有操作系统

每一种硬件都要有自己的驱动程序---操作系统的触手

计算机操作系统层状

为啥有OS，对下管理好所有的软硬件资源

对上给用户提供良好的运行环境

操作系统是如何管理的？（顶层理解）

操作系统就是校长

底层硬件就是学生

（驱动程序）中间角色辅导员----对上获取数据

对下---执行决策工作

操作系统是如何管理硬件的呢

Struct device

{

Name;

Status;

……

}

在操作系统内部实现了device\_list 的单链表

对硬件的管理：对链表的增删查改

这种管理思路叫—先描述再组织：想要管理任何一个系统，都要先把被管理的对象描述出来，把所有的被管理对象用某种数据结构组织起来---所有计算机对象，管理的思路都遵守该原则

所以：为什么现在所有的主流的面向对象语言都要提供：1，面向对象能力2，标准库，如C++?

1. 给予先描述的能力
2. 给与再组织都能力

为什么会存在数据结构这么学科？？

为什么现在主流的是面向对象？

因为可以更好的描述对象

程序不能直接向硬件写入，因为操作系统管理硬件，程序不能绕过硬件向显示器写入

操作系统必须提供各种接口，方便上层使用---操作系统开放只开放系统调用---银行窗口

操作系统用C语言写的对上的接口也只能是c语言的接口

有没有不会使用这些窗口的人？---》搞一个大堂经理（服务层）

---系统调用接口需要对系统有一定的了解，对一般程序员，使用比较麻烦----将系统调用接口封装一些库----》如glibc,libc++库

有些库需要封装接口，有些库不需要不需要通过操作系统比如---数学库---使用CPU的资源

显示器就是一个二维的·坐标

安卓的底层也是linux---

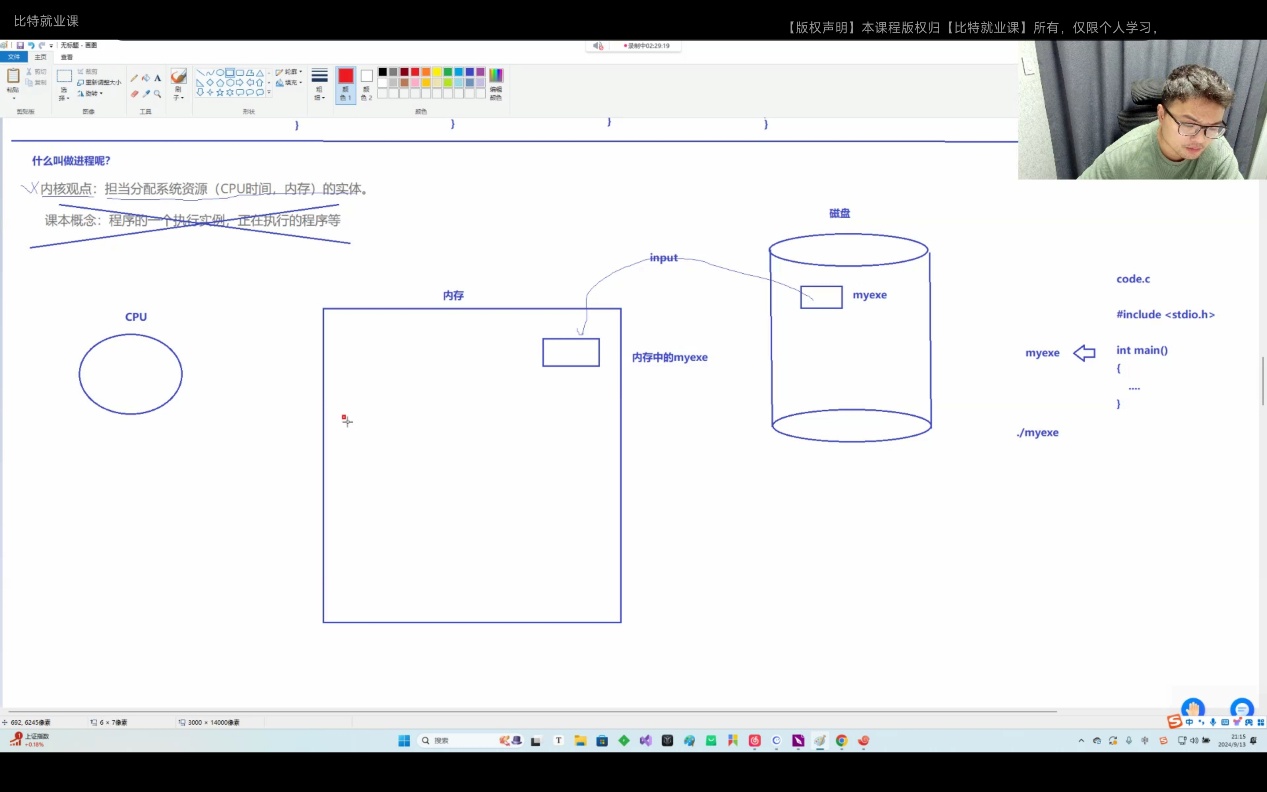
进程

进程管理---先描述再组织

什么叫做进程：

进程：

根据冯诺依曼体系：我们计算机中的一个程序实际上是磁盘上的一个文件



Input 过程由操作系统调用加载器完成执行

操作系统属于软件，是计算机里启动的第一个程序

操作系统通过进程的属性数据管理进程

Linux 叫做任务结构体Struct task\_struct---课本上叫做进程控制块

任务结构体里会包含----pid进程id(int pid)---进程状态(int status)---进程优先级(int prio)---进程代码和数据的地址(void\*memptr)-- 内存指针---上下文：Struct task\_struct \*next;

进程=内核数据结构（task\_struct）+程序的代码和数据

学生=人（学校）+属性（系统）

为什么要有进程？？

因为操作系统要有PCB啊—因为os要管理进程，先描述，再组织