北京大学 人工智能研究院 INSTITUTE FOR ARTIFICIAL INTELLIGENCE, PEKING UNIVERSITY
北京大学元培学院 YUANPEI COLLEGE PEKING UNIVERSITY

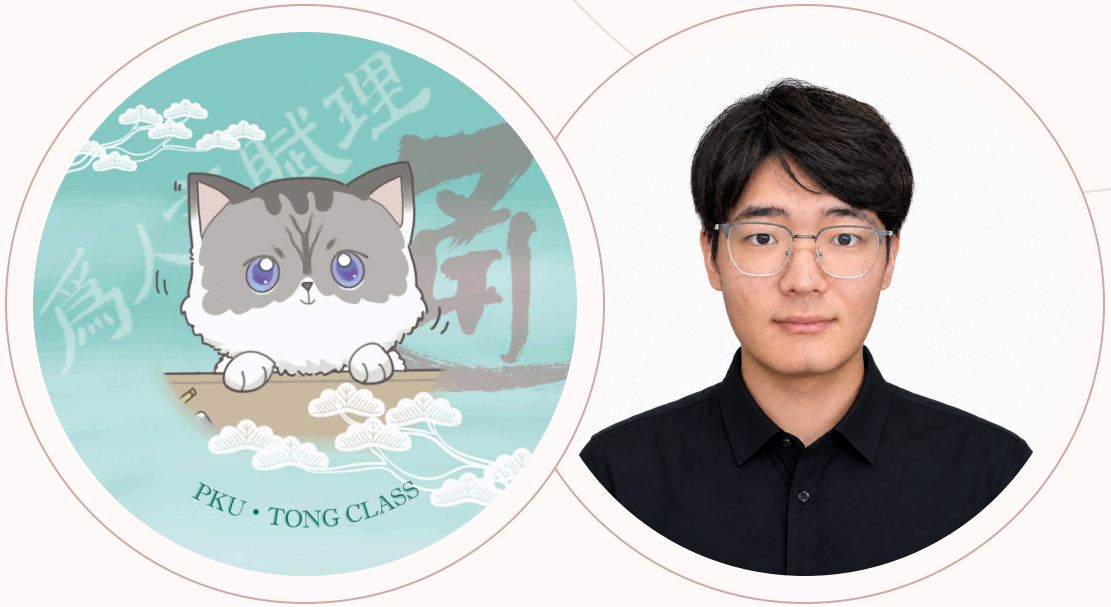
ToNG: Tutorials on Necessary Groundwork

通班人工智能科研先导课 · 第二讲

操作系统与 Linux

Operating Systems and Linux

—— How Programs Meet the Machine



主讲单位
北京大学通用人工智能实验班

主讲人
石伟业

— 2.1 学习目标

这节课的主线任务

系统

理解操作系统

从系统层级的资源管理出发，看 CPU、内存、磁盘和网络如何被组织。

环境

分清系统环境

分清物理机与硬件、操作系统、软件、虚拟机等之间的层次。

排查

找到报错原因

从资源、位置、权限入手，再用命令验证判断。

小贴士：命令的语法可以随时查询（参见第零讲），处理操作系统的思维方式更重要。

— 2.2 前言

硬件不会替程序排队

CPU 和 GPU、RAM 和磁盘已经为计算和存储提供了资源，但当多个程序同时请求它们时，系统还必须回答三个问题：

资源怎么分？ · 资源在哪里？ · 分配的边界？

谁来协调这些复杂的请求，又是谁替程序的安全运行划出边界？

— 2.3 操作系统是什么

操作系统让多任务有序，又有边界

01

协调资源

协调 CPU、内存和磁盘，安排任务调度。

02

划定边界

不同程序的运行资源进行隔离。

03

提供抽象

上层用文件和进程组织资源。

接下来：沿着一次“读文件”请求，看系统具体接过了什么工作。

— 示例 一条请求的层次

“读文件”请求如何到达硬件

- 应用只发送目标：`open("notes.txt")`。
- 系统先确认文件位置、存在性和读写权限，再安排资源。
- 最后由存储、CPU、内存等硬件完成真正的读写和计算。

那么，操作系统内部是怎么分工的呢?

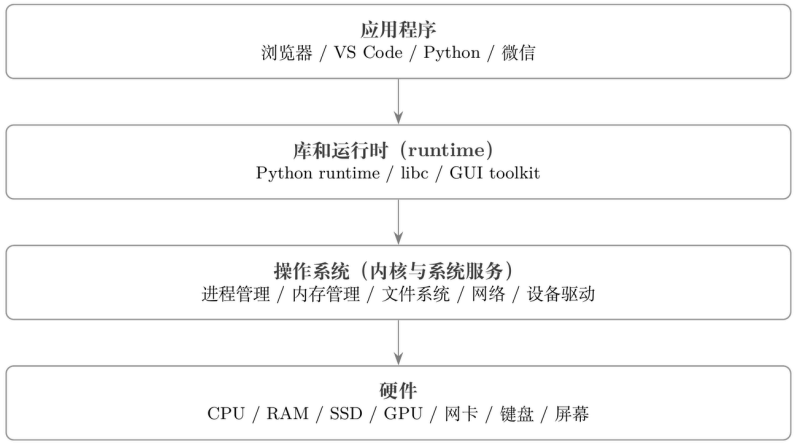


图 2.1: 计算机系统的四层视图：上层软件通过下层提供的能力完成工作。

— 2.4 Linux 与 Linux 发行版

Linux 操作系统内部的层次

◆ 底层

Kernel

Linux 是最核心的内核部分

它负责直接对接资源，处理进程、内存、文件系统、网络和设备等核心事务；桌面和终端里的应用最终都通过它请求资源。

◆ 完整版

Distribution

发行版是包含内核的完整操作系统

它负责对接用户，提供更丰富的桌面、软件和管理工具。常见 Linux 发行版有 Ubuntu，Fedora，Red Hat 等。

— 2.5 虚拟机

虚拟机：让系统在沙盒中运行

- 宿主机提供资源；虚拟化软件把资源呈现给客户机。
- 客户机里的 Ubuntu 像独立电脑一样运行。
- 环境与系统和其他虚拟机隔离、可回退、可重装。

注意：虚拟机不会凭空增加 CPU、内存或磁盘等硬件资源，只是通过虚拟化技术把宿主机的这些资源定向到虚拟机中

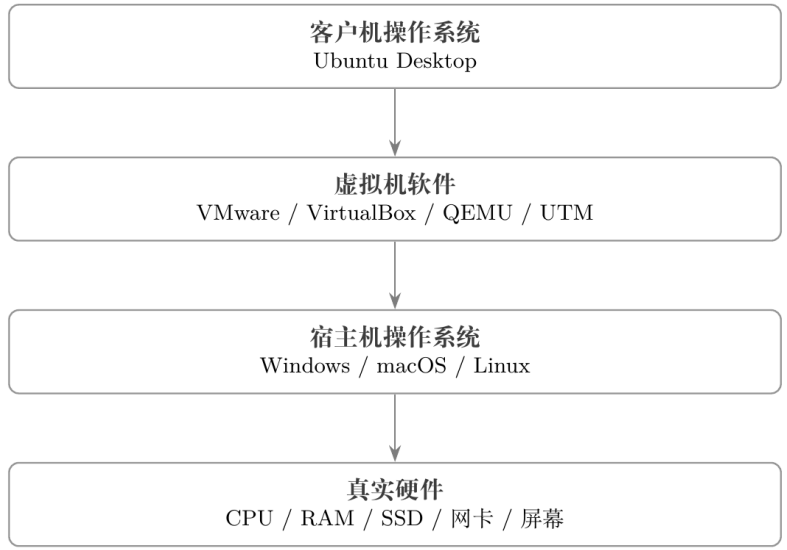


图 2.4: 虚拟机的分层结构：客户机系统通过虚拟机软件使用宿主机的资源。

— 2.5 虚拟机资源配置

给客户机资源，也给宿主机留余量

4-8 GB

内存

客户机能运行，也别挤占宿主机资源。

2-4 cores

CPU

让客户机能响应；虚拟核最终仍在真实 CPU 上排队。

25 GB+

磁盘

系统和项目都会占用虚拟磁盘空间。

安装虚拟机时，除了分配硬件资源，还需要安装操作系统对应的软件。对于 Linux 这种开源系统，可以从发行版的官网下载系统镜像，加载到虚拟机软件中进行安装。

环境准备好后：打开终端，输入 help，了解基本的命令和使用方式。

— 2.6 操作系统基础知识

一条命令如何被系统执行

- **shell** 理解命令，找到解释器和脚本。
- **操作系统** 检查路径、权限，准备进程和内存，读取所需文件。
- **CPU 与 I/O** 执行指令，把结果送到屏幕、文件或网络。

同样的程序在不同机器上运行结果不同，往往是某一步出了问题。

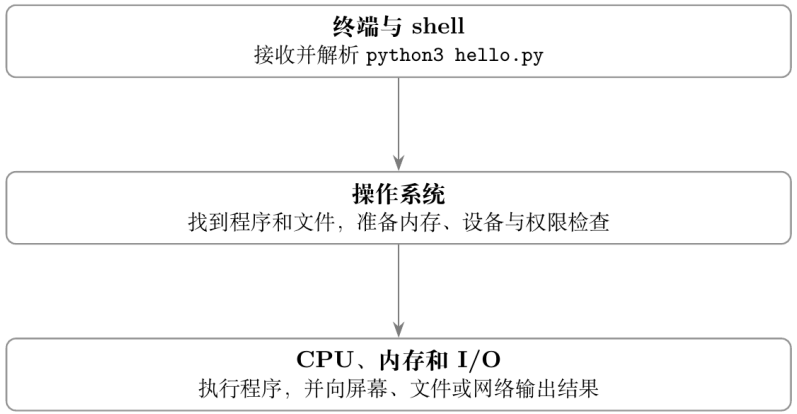


图 2.5: 一条终端命令背后的基本协作关系。

— 2.6 操作系统基础知识

程序访问关键资源，必须通过内核

用户态

User space

应用在这里运行

应用获得操作系统分配的资源，可以执行计算任务，也可以向有权限写入的路径写入文件，但不能直接改写别的进程或占用分配给其他进程的资源，也不能未经授权访问其他硬件。

内核态

Kernel space

内核掌握关键资源

应用程序打开文件、建进程、申请内存等等请求，都要通过内核处理，在合理的调度下按顺序分配有限资源。内核是操作系统的核心，一旦内核发生故障整个操作系统都将停止运行。

— 2.6 分清资源类型

RAM、磁盘和显存：不能互相替代

RAM

临时工作区

正在运行的代码和数据先放在这里；断电即清空，容量小而速度快。

磁盘

长期仓库

永久存储，容量大，但不能直接和 CPU、GPU 交互，速度也较慢

VRAM

GPU 工作区

GPU 计算时放模型和数据；普通 RAM 很多也不代表这里够用。

虚拟内存技术：当系统内存严重不足时，操作系统会在提前预留好的磁盘区域开设一部分作为“虚拟内存”使用来维持系统正常运行。由于数据被换到更慢的磁盘，系统就会明显变慢，这时要排查具体是哪一个进程在占资源。

— 检查运行中的程序

程序启动后，系统将它作为进程管理

Program

即程序，是存储在磁盘上的脚本或可执行文件，是一份静态文件。

Process

即进程，是实际运行中的对象。进程启动后，系统会追踪 PID（进程的编号）、用户以及各项资源占用。

在 Linux 中，可以用 `top` 看总体的资源占用情况，`free -h` 看内存消耗，`df -h` 看磁盘占用。
遇到资源紧张或系统卡顿问题时，应该先观察系统状态，找出消耗资源的“魁首”，再解决问题。

如果资源没有问题：再确认权限管理等等是否有问题

— 2.7 常用 Bash 命令

◆ 路径相关指令

路径先行：确认工作范围

Bash

```
pwd                 # 我现在在哪个目录？是英文 print working directory 的缩写
ls                  # 这里有什么？是英文 list 的缩写，可用参数 -a 显示包含隐藏的文件的所有文件
cd /path/to/dir     # 去别的目录！是英文 change directory 的缩写，其中 / 是根目录，~ 是用户的 home
                    # . 和 .. 分别代表当前目录和当前目录的上一级目录
```

相对路径总从当前目录开始；`/home/tong/notes.txt` 的起点固定，`./notes.txt` 的含义则会随当前位置变化。

位置确定后：系统还要判断你有没有权限读、写或执行这个目标。

— 2.7 常用 Bash 命令

◆ 怎样执行命令

文件存在≠直接执行

Bash

```
bash run.sh          # 明确要求 Bash 读取脚本
./run.sh             # 请求系统直接执行脚本，可以直接运行 shell 脚本，python 脚本等
chmod +x run.sh      # 添加执行权限，那之后就可以用 ./ 运行
```

下面是在 Bash 终端中查看该文件权限时，会看到的提示符和输出：

```
tongtong@ubuntu:~/ToNG$ ls -la run.sh
-rwxr-xr-- 1 tongtong tongclass 678 Aug 12 19:13 run.sh
```

其中 `r`、`w`、`x` 分别表示 read、write、execute 权限；`tongtong` 是文件所有者，`tongclass` 是所属用户组。直接执行脚本时，还需要在脚本首行写明解释器，例如 `#!/bin/bash` 表示使用 Bash 执行。

— 2.7 常用 Bash 命令

◆ 数据去了哪里?

挂载和管道：数据进入和传递

Bash

```
echo meow > kitty.txt
ps aux | grep python
```

程序输出	文件或下一个程序	新的结果
stdout	`>` 或 `|`	继续显示或保存

U 盘、额外磁盘和共享文件夹要先挂载到目录树；`>` 改变输出去向，`|` 把前一个程序的输出交给下一个程序。

—— 排查问题的三个方向 ——

遇到问题，先判断资源、位置和权限

资源

它在等什么?

CPU、RAM、磁盘、网络，还是某一个持续运行的进程?

位置

它从哪里找?

当前目录、完整路径和挂载点，起点是不是正确?

权限

系统为什么拒绝?

检查用户身份、读写执行权限和解释器条件。

先明确问题，再用一条命令验证最可能的原因。这样排查时就不会盲目试命令。