应用程序眼中的操作系统

蒋炎岩

南京大学 | 计算机软件研究所 | 系统与软件分析研究组

操作系统和应用程序之间的约定

用户、应用程序和操作系统

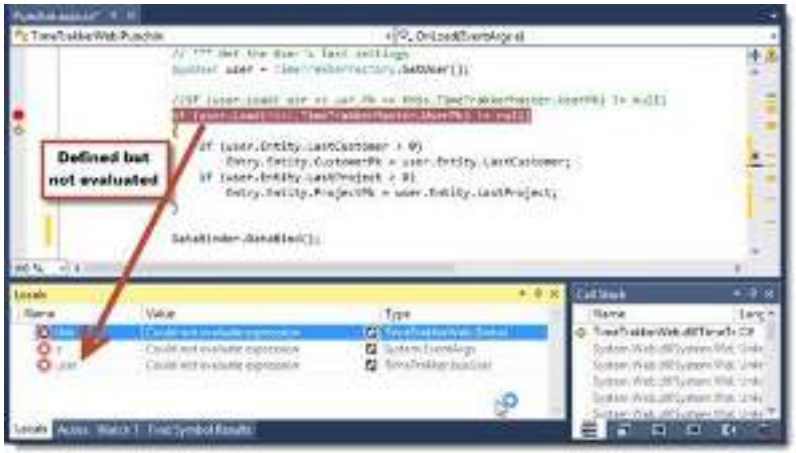
- 用户：我要XXX
- 应用程序：为用户提供XXX
- 操作系统：为应用程序实现XXX提供支持
- XXX = 发微信、玩游戏、用外挂、杀病毒、写代码、调程序、查看/改变系统状态……

操作系统和应用程序之间的约定

- 需要定义操作系统中的对象
- 以及规定程序能对对象做何种操作
- 系统调用：对指定的对象进行操作

操作系统中的对象

- 讨论：如果希望支持各类应用程序，操作系统里应该有哪些对象、哪些操作？



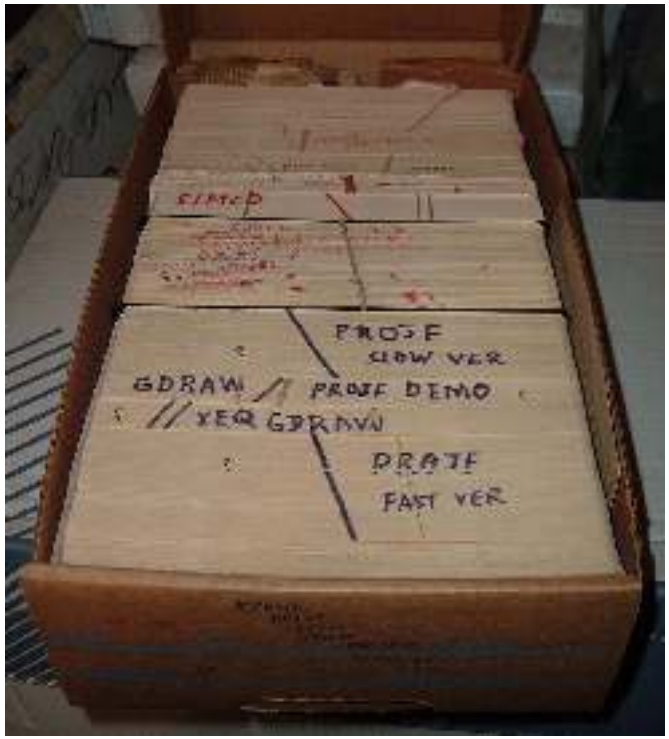
操作系统中的对象：执行的程序

- 操作系统能支持多个程序执行
- 执行的程序称为“进程”
  - 拥有自己的内存，好像独占CPU
- 进程之间可以交互
  - 发送消息、传递信息
  - 查看进程的状态(读取内存)
  - 改变进程的状态(如杀死进程)

操作系统中的对象：数据

- 回到1950s：批处理时代，计算机分批运行程序
  - 读取输入 → 计算 → 输出

- “文件”被载入计算机运行
  - IBM 350的整个磁盘是一个“文件”
  - 文件 = 一些数据
  - 操作：读、写、改名、……



文件系统

- 当存储器(磁盘)容量变大，可以存储更多数据了
  - 能够存放多个“文件” (1961)
  - 每个文件有自己的名字
  - 用相应的库函数，通过名字访问文件

- 存储器变得更大
  - 能存放成千上万个文件，自然就以目录的形式组织起来
    - 参考图书馆的设计
    - “C:\Program Files”、“/etc”.....

一个有趣的UNIX设计

- 文件 = 一些数据 (文件在1950s时代的定义)
- 计算机系统里的一切 = 一些数据 (ICS课程已经学过)
- 推论：计算机系统里的一切 = 文件 (纳尼？)

- Everything is a file
  - 系统的状态、进程间通信的连接、进程的内存、I/O设备……它们都可以看成是文件
  - 文件 = 一些数据(和数据上的操作)
    - 读(read)、写(write)、定位(lseek)

Everything is a File：演示

- 数据是文件
- 磁盘是文件
- 系统状态是文件
- 输入流是文件
- 进程的内存是文件

> UNIX was not designed to stop you from doing stupid things, because that would also stop you from doing clever things.
>
> — *Doug Gwyn*

- Linux里的一些虚拟文件系统：
  - devfs – Device FS：设备皆文件
  - procfs – Process FS：进程皆文件
  - sysfs – System FS：系统皆文件

业务操作员，应用后台home目录下
`rm -rf ./*XXXX*`
不知怎么敲成了`rm -rf ./* XXXX*`
多了个空格

X，TMD，这可是生产环境啊，完了，出事了

匿名用户 @ 知乎
类似事情发生在Gitlab

这就(大约)是这门课的全部了

- 进程：拥有虚拟的处理器和地址空间的程序
- 文件：操作系统中的一切数据
- 讨论：两种实现
  - UNIX：Everything is a File
  - Windows：操作系统提供各种API
  - 它们总体来说能实现的功能是等价的(都能为应用程序提供服务)，两种做法做出了怎样的取舍？

Busybox眼中的操作系统

Busybox：一系列的应用程序

- 命令行工具的集合体
  - busybox vi
  - busybox httpd
  - busybox ash
  - busybox grep
  - ...
  - 大部分工具都是Linux发行版已经集成的
  - Android系统自带了busybox (只需终端模拟器就能启动)
- 它代表了各种应用程序
  - 是个25万行代码的小项目

拿到项目你就懵逼了

- 下载了一个源代码包.tar.bz2
- 我就一个命令行，我到底做什么啊？？？
  - 解压缩
  - 编译(make)
  - 执行

- 这是操作系统和用户之间的约定
  - “Unix is *user-friendly* — it‘s just choosy about who its friends are.”

Busybox源代码导读

- 每一个程序都是一个小工具
  - 比如vi_main(int argc, char **argv);
- 仅有一个入口
  - appletlib.c:976 (怎么找到它的?)
  - 执行一系列初始化
  - 调用run_applet_and_exit(applet_name, argv)
    - 查表(include/applet_tables.h)，但是这个文件刚开始是不存在的(?)
    - 最终调用xxx_main

进入Busybox源代码

- 讲解
  - procps/uptime.c
  - coreutils/link.c
  - coreutils/rm.c

- 大家可以选自己有兴趣的程序阅读
  - 总会对一些命令的实现有兴趣，不然也许转专业更适合你
  - RTFM & RTFSC：busybox的功能和命令行工具基本兼容，可以直接使用man查看功能，然后阅读相应的源代码
  - 读代码没有想象的那么难——建立起基本概念，学习常见的模式

Busybox与操作系统的交互

- strace工具提供了我们观察程序执行的手段
  - 从execve(“./busybox”, [“./busybox”, “true”], [/* 46 vars */])开始
  - 执行加载过程(多次access，然后是libm和libc)
  - 创建基本的运行环境(set_thread_area、mprotect)
  - (实际执行main函数)
  - 以exit_group(0)结束