



南京大學
NANJING UNIVERSITY



可执行文件的加载

南京大学

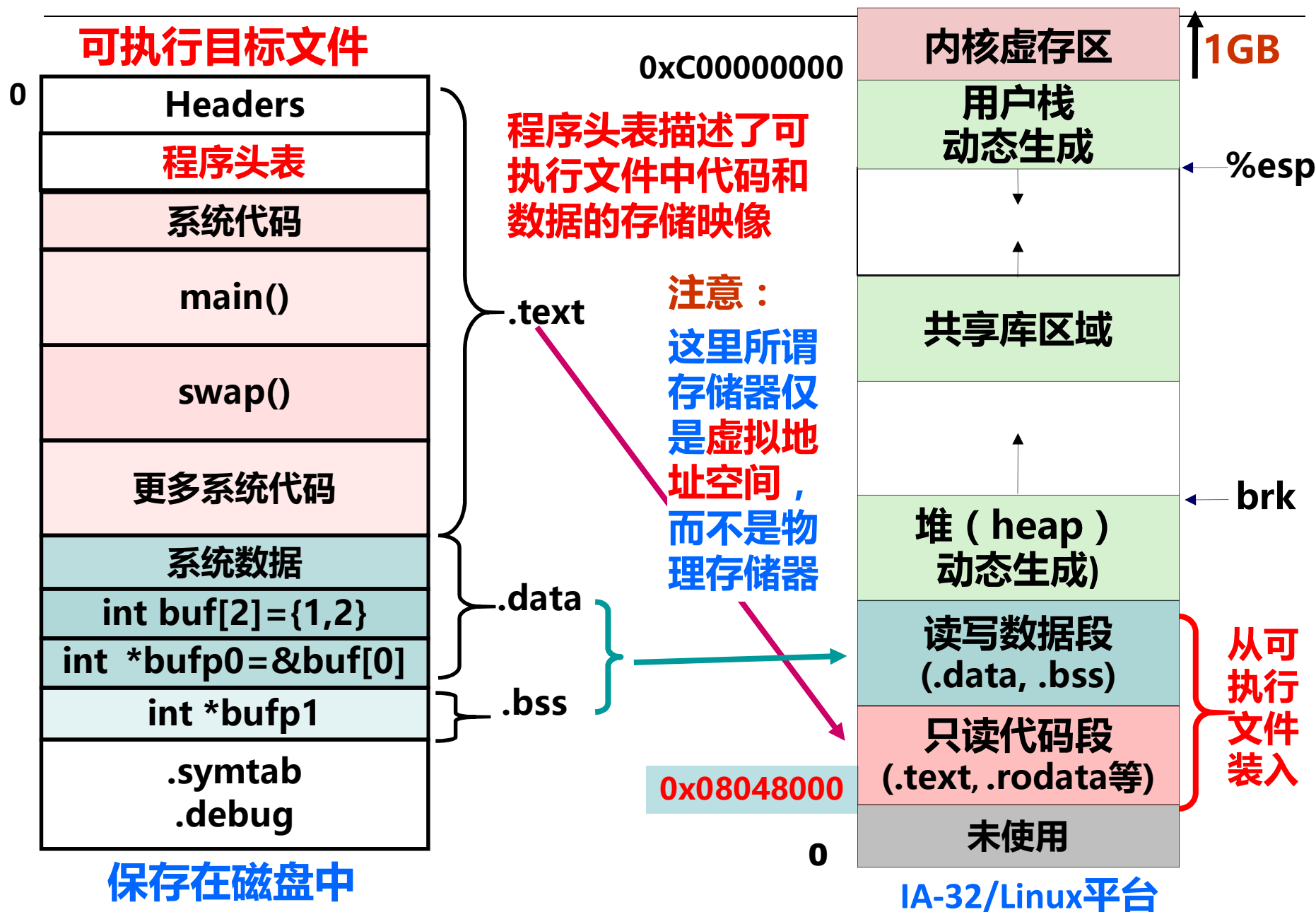
计算机科学与技术系

袁春风

email: cfyuan@nju.edu.cn

2015.6

可执行文件的存储器映像



可执行文件的加载

- 通过调用execve系统调用函数来调用加载器
- 加载器（loader）根据可执行文件的程序（段）头表中的信息，将可执行文件的代码和数据从磁盘“**拷贝**”到存储器中（**实际上不会真正拷贝，仅建立一种映射，这涉及到许多复杂的过程和一些重要概念，将在后续课上学习**）
- 加载后，将PC（EIP）设定指向Entry point（即符号_start处），最终执行main函数，以启动程序执行

程序被启动
如 **\$./hello**

调用fork()

以构造的argv和envp
为参数调用execve()

execve()调用加载器
进行可执行文件加载，
并最终转去执行main

_start: __libc_init_first → _init → atexit → main → _exit

ELF头信息举例

[BACK](#)

\$ readelf -h main

ELF Header:

Magic: **7f 45 4c 46** 01 01 01 00 00 00 ...

Class: ELF32

Data: 2's complement, little endian

Version: 1 (current)

OS/ABI: UNIX - System V

ABI Version: 0

Type: EXEC (Executable file)

Machine: Intel 80386

Version: 0x1

Entry point address: **x8048580**

Start of program headers: 52 (bytes into file)

Start of section headers: 3232 (bytes into file)

Flags: 0x0

Size of this header: 52 (bytes)

Size of program headers: 32 (bytes)

Number of program headers: 8

Size of section headers: 40 (bytes)

Number of section headers: 29

Section header string table index: 26

注意：程序入口地址并不是0x8048000

ELF 头
程序头表
.init 节
.text 节
.rodata 节
.data 节
.bss 节
.symtab 节
.debug 节
.strtab 节
.line 节
Section header table (节头表)

只读代码段

程序的加载和运行

- UNIX/Linux系统中，可通过调用execve()函数来启动加载器。
- execve()函数的功能是在当前进程上下文中加载并运行一个新程序。
execve()函数的用法如下：

```
int execve(char *filename, char *argv[], *envp[]);
```

filename是加载并运行的可执行文件名(如./hello)，可带参数列表argv和环境变量列表envp。若错误（如找不到指定文件filename），则返回-1，并将控制权交给调用程序；若函数执行成功，则不返回，最终将控制权传递到可执行目标中的主函数main。

- 主函数main()的原型形式如下：

```
int main(int argc, char **argv, char **envp); 或者：
```

```
int main(int argc, char *argv[], char *envp[]);
```

argc指定参数个数，参数列表中第一个总是命令名（可执行文件名）

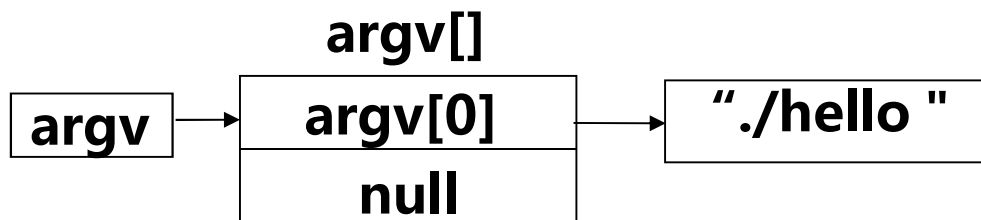
例如：命令行为“ld -o test main.o test.o”时，argc=6

程序的加载和运行

问题：hello程序的加载和运行过程是怎样的？

Step1：在shell命令行提示符后输入命令：`$/./hello[enter]`

Step2：shell命令行解释器构造argv和envp



[BACK](#)

Step3：调用**fork()**函数，创建一个子进程，与父进程shell完全相同（只读/共享），包括只读代码段、可读写数据段、堆以及用户栈等。

Step4：调用**execve()**函数,在当前进程（新创建的子进程）的上下文中加载并运行hello程序。将hello中的.text节、.data节、.bss节等内容加载到当前进程的虚拟地址空间（仅修改当前进程上下文中关于存储映像的一些数据结构，不从磁盘拷贝代码、数据等内容）

Step5：调用hello程序的**main()**函数，hello程序开始在一个进程的上下文中运行。
`int main(int argc, char *argv[], char *envp[]);`