姓名：周仙辉

日期：2024年3月19日

程序执行中函数的调用过程

函数调用

1.将call的下一条指令地址压栈。

2.保存实参，64位通过寄存器（当数量过多再用栈），32位通过栈

3.跳转到函数体

函数体执行

4.开辟栈帧，如果函数体有临时变量，将变量压栈

5.执行函数体

6.销毁栈帧

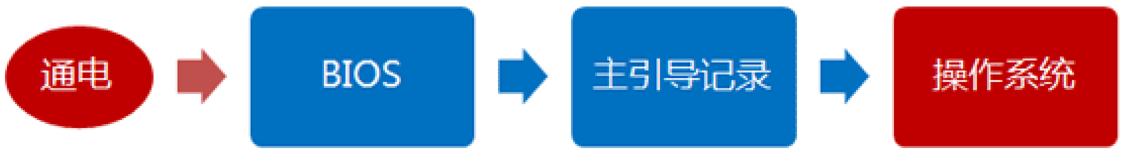
返回

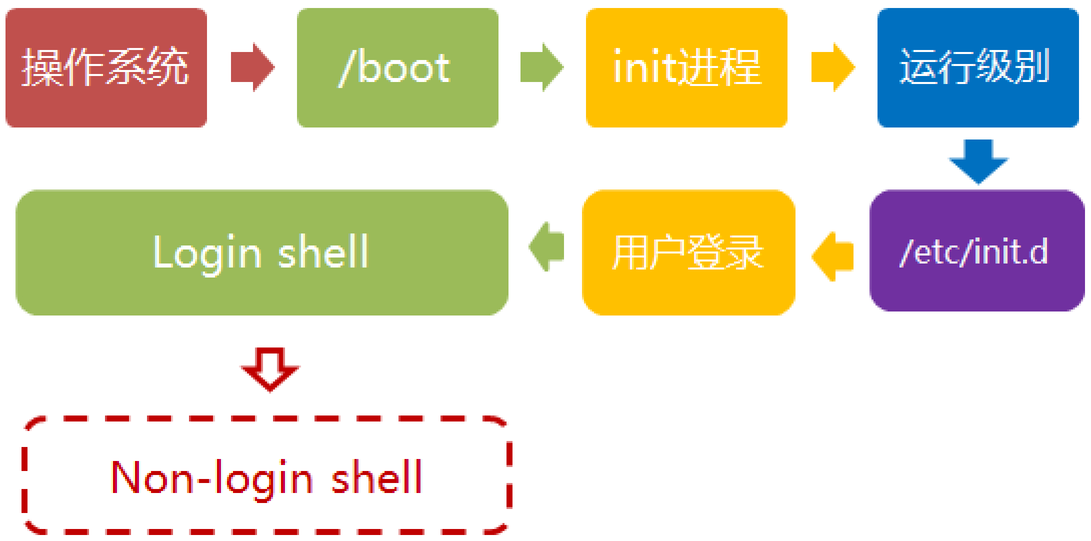
7.将开始压栈的地址出栈。

linux启动过程

看了这篇文章Linux 的启动流程 - 阮一峰的网络日志 (ruanyifeng.com)和计算机是如何启动的？ - 阮一峰的网络日志 (ruanyifeng.com)

大致为一下几个过程：





2.1、2.2、2.3、3.4、2.5、2.6、2.7、2.8、2.9

2.1 What is the purpose of system calls?

System calls allow user-level processes to request services of the operating system.

2.2 What are the five major activities of an operating system with regard to process management?

The five major activities are:

a. The creation and deletion of both user and system processes

b. The suspension and resumption of processes

c. The provision of mechanisms for process synchronization

d. The provision of mechanisms for process communication

e. The provision of mechanisms for deadlock handling

2.3 What are the three major activities of an operating system with regard to memory management?

The three major activities are:

a. Keep track of which parts of memory are currently being used and by whom.

b. Decide which processes are to be loaded into memory when memory space becomes available.

c. Allocate and deallocate memory space as needed.

2.4 What are the three major activities of an operating system with regard to secondary-storage management?

The three major activities are:

• Free-space management.

• Storage allocation.

• Disk scheduling.

2.5 What is the purpose of the command interpreter? Why is it usually separate from the kernel?

It reads commands from the user or from a file of commands and creates processes to execute them, usually by turning them into one or more system calls. It is usually not part of the kernel since the command interpreter is subject to changes.

2.6 What system calls have to be executed by a command interpreter or shell in order to start a new process?

fork 然后 execve

2.7 What is the purpose of system programs?

System programs can be thought of as bundles of useful system calls. They provide basic system-level functionality to users so that users do not need to write their own programs to solve common problems.

2.8 What is the main advantage of the layered approach to system design? What are the disadvantages of the layered approach?

The main advantage of the layered approach is modularity. The layered approach is selected such that each layer uses function and services of only lower level layers. This approach simplifies debugging and system verification.

The problem is deciding what order in which to place the layers, as no layer can call upon the services of any higher layer, and so many chicken-and-egg situations may arise.

Layered approaches can also be less efficient, as a request for service from a higher layer has to filter through all lower layers before it reaches the HW, possibly with significant processing at each step.

2.9 List five services provided by an operating system, and explain how each creates convenience for users. In which cases would it be impossible for user-level programs to provide these services? Explain your answer.

The five services are:

a. Program execution. The operating system loads the contents (or sections) of an executable file into memory and begins its execution. A user-level program could not be trusted to properly allocate CPU time.

b. I/O operations. Disks, tapes, serial lines, and other devices must be communicated with at a very low level. The user need only specify the device and the operation to perform on it, while the system converts that request into device-specific or controller-specific commands. User-level programs cannot be trusted to access only devices which they should have access to and to access them only when they are otherwise unused.

c. File-system manipulation. There are many details in file creation, deletion, allocation, and naming that users should not have to perform. Blocks of disk space are used by files and must be tracked. Deleting a file requires removing the name file information and freeing the allocated blocks. Protections must also be checked to assure proper file access. User programs could neither ensure adherence to protection methods nor be trusted to allocate only free blocks and deallocate blocks on file deletion.

d. Communications. Message passing between systems requires messages to be turned into packets of information, sent to the network controller, transmitted across a communications medium, and reassembled by the destination system. Packet ordering and data correction must take place. Again, user programs might not coordinate access to the network device, or they might receive packets destined for other processes.

e. Error detection. Error detection occurs at both the hardware and software levels. At the hardware level, all data transfers must be inspected to ensure that data have not been corrupted in transit. All data on media must be checked to be sure they have not changed since they were written to the media. At the software level, media must be checked for data consistency; for instance, whether the number of allocated and unallocated blocks of storage match the total number on the device. There, errors are frequently process-independent (for instance, the corruption of data on a disk), so there must be a global program (the operating system) that handles all types of errors. Also, by having errors processed by the operating system, processes need not contain code to catch and correct all the errors possible on a system.

2.12、2.13、2.15、2.16、2.17、2.18、2.19、2.21、2.22

2.12 The services and functions provided by an operating system can be divided into two main categories. Briefly describe the two categories, and discuss how they differ.

Functions and services which are directly **helpful to the user** (networking, computer sharing, error detection, consistent computing)

Functions and resources which ensure efficient operation through resource sharing; resource allocation, user account tracking, and protection/security

2.13 Describe three general methods for passing parameters to the operating system.

a. Pass parameters in registers

b. Registers pass starting addresses of blocks of parameters

c. Parameters can be placed, or pushed, onto the stack by the program, and popped off the stack by the operating system

2.15 What are the five major activities of an operating system with regard to file management?

a. The creation and deletion of files

b.The creation and deletion of directories

c.The support of primitives for manipulating files and directories

d.The mapping of files onto secondary storage

e.The backup of files on stable (nonvolatile) storage media

2.16 What are the advantages and disadvantages of using the same system-call interface for manipulating both files and devices?

advantages:

Each device can be accessed as though it was a file in the file system. Since most of the kernel deals with devices through this file interface, it is relatively easy to add a new device driver by implementing the hardware-specific code to support this abstract file interface. Therefore, this benefits the development of both user program code, which can be written to access devices and files in the same manner, and device-driver code, which can be written to support a well-defined API.
disadvantages:

The disadvantage with using the same interface is that it might be difficult to capture the functionality of certain devices within the context of the file access API, thereby resulting in either a loss of functionality or a loss of performance. Some of this could be overcomed by the use of the ioctl operation that provides a general-purpose interface for processes to invoke operations on devices.

2.17 Would it be possible for the user to develop a new command interpreter using the system-call interface provided by the operating system?

可以

命令解释器允许用户这样创建和管理进程并且确定它们的通信方式(通过管道和文件)。由于所有这些功能都可以被用户级的程序进行系统调用，因此用户可以开发出一个新的命令行解释器。

2.18 What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?

Answer:

One is message-passing model and the other is shared-memory model.

Shared-memory strengths and weaknesses: it allows higher speed and convenience of communication. However, in the areas of protection and synchronization between the processes some problems exist.

Message-passing strengths and weaknesses: message can be exchanged between the processes either directly or indirectly through a common mailbox. It is useful for exchanging smaller amounts of data and easier to implement for inter-computer communication. However, its speed is slower than shared-memory model.

2.19 Why is the separation of mechanism and policy desirable?

Mechanism and policy must be separate to ensure that systems are easy to modify.

No two system installations are the same, so each installation may want to tune the operating system to suit its needs. With mechanism and policy separate, the policy may be changed at will while the mechanism stays unchanged. This arrangement provides a more flexible system.

2.21 What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?

Benefits typically include the following:

(a) adding a new service does not require modifying the kernel,

(b) it is more secure as more operations are done in user mode than in kernel mode.

A simpler kernel design and functionality typically results in a more reliable operating system. User programs and system services interact in a microkernel architecture by using inter process communication mechanisms such as messaging. These messages are conveyed by the operating system.

The primary disadvantages of the microkernel architecture are the overheads associated with inter process communication and the frequent use of the operating system's messaging functions in order to enable the user process and the system service to interact with each other (context switching).

2.22 What are the advantages of using loadable kernel modules?

Without loadable kernel modules, an operating system would have to include **all possible anticipated functionality already compiled directly into the base kerne**l. Much of that functionality would reside in memory without being used, wasting memory, and would require that users rebuild and reboot the base kernel every time they require new functionality. Most operating systems supporting loadable kernel modules will include modules to support most desired functionality, and **dynamically** load the modules as needed.

可以追踪程序执行的函数调用的工具

已知的就是gdb，这是官方文档