main(argc, argv, envp)

其中argc是该命令行内有关参数数目的计数器，包括程序名称。例如，上面的例子中，argc为3。

第二个参数argv是一个指向数组的指针。该数组的元素i是指向该命令行第i个字串的指针。在本例中，argv[0]指向字串“cp”，argv[1]指向字符串“file1”，argv[2]指向字符串“file2”。

main的第三个参数envp指向环境的一个指针，该环境是一个数组，含有name = value的赋值形式，用以将诸如终端类型以及根目录等信息传送给程序。还有供程序可以调用的库过程，用来取得环境变量，这些变量通常用来确定用户希望如何完成特定的任务（例如，使用默认打印机）。在图1-19中，没有环境参数传递给子进程，所以execve的第三个参数为零。

如果读者认为exec过于复杂，那么也不要失望。这是在POSIX的全部（语义上）系统调用中最复杂的一个，其他的都非常简单。作为一个简单例子，考虑exit，这是在进程完成执行后应执行的系统调用。这个系统调用有一个参数，退出状态（0至255），该参数通过waitpid系统调用中的statloc返回给父进程。

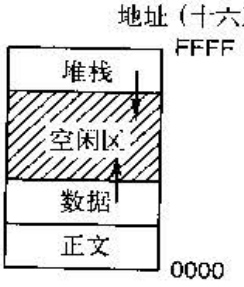

图1-20 进程有三段：正文段、数据段和堆栈段

在UNIX中的进程将其存储空间划分为三段：正文段（如程序代码）、数据段（如变量）以及堆栈段。数据段向上增长而堆栈向下增长，如图1-20所示。夹在中间的是未使用的地址空间。堆栈在需要时自动地向中间增长，不过数据段的扩展是显式地通过系统调用brk进行的，在数据段扩充后，该系统调用指定一个新地址。但是，这个调用不是POSIX标准中定义的调用，对于存储器的动态分配，我们鼓励程序员使用malloc库过程，而malloc的内部实现则不是一个适合标准化的主题，因为几乎没有程序员直接使用它，我们有理由怀疑，会有什么人注意到brk实际不是属于POSIX的。

1.6.2 用于文件管理的系统调用

许多系统调用与文件系统有关。本小节讨论在单个文件上的操作，1.6.3节将讨论与目录和整个文件系统有关的内容。

要读写一个文件，先要使用open打开该文件。这个系统调用通过绝对路径名或指向工作目录的相对路径名指定要打开文件的名称，而代码O_RDONLY、O_WRONLY或O_RDWR的含义分别是读、写或两者。为了创建一个新文件，使用O_CREAT参数。然后可使用返回的文件描述符进行读写操作。接着，可以用close关闭文件，这个调用使得该文件描述符在后续的open中被再次使用。

毫无疑问，最常用的调用是read和write。我们在前面已经讨论过read。write具有与read相同的参数。

尽管多数程序频繁地读写文件，但是仍有一些应用程序需要能够随机访问一个文件的任意部分。与每个文件相关的是一个指向文件当前位置的指针。在顺序读（写）时，该指针通常指向要读出（写入）的下一个字节。lseek调用可以改变该位置指针的值，这样后续的read或write调用就可以在文件的任何地方开始。

lseek有三个参数：第一个是文件的描述符，第二个是文件位置，第三个说明该文件位置是相对于文件起始位置、当前位置，还是文件的结尾。在修改了指针之后，lseek所返回的值是文件中的绝对位置。

UNIX为每个文件保存了该文件的类型（普通文件、特殊文件、目录等），大小，最后修改时间以及其他信息。程序可以通过stat系统调用查看这些信息。第一个参数指定了要被检查的文件；第二个参数是一个指针，该指针指向用来存放这些信息的结构。对于一个打开的文件而言，fstat调用完成同样的工作。

1.6.3 用于目录管理的系统调用

本节我们讨论与目录或整个文件系统有关的某些系统调用，而不是1.6.2节中与一个特定文件有关的系统调用。mkdir和rmdir分别用于创建和删除空目录。下一个调用是link。它的作用是允许同一个文件以两个或多个名称出现，多数情形下是在不同的目录中这样做。它的典型应用是，在同一个开发团队中允许若干个成员共享一个共同的文件，他们之中的每个人都在自己的目录中有该文件，但可能采用的是不同的名称。共享一个文件，与每个团队成员都有一个私用副本并不是同一件事，因为共享文件意味着，任何成员所做的修改都立即为其他成员所见——只有一个文件存在。而在复制了一个文件的多个副本之后，对其中一个副本所进行的修改并不会影响到其他的副本。