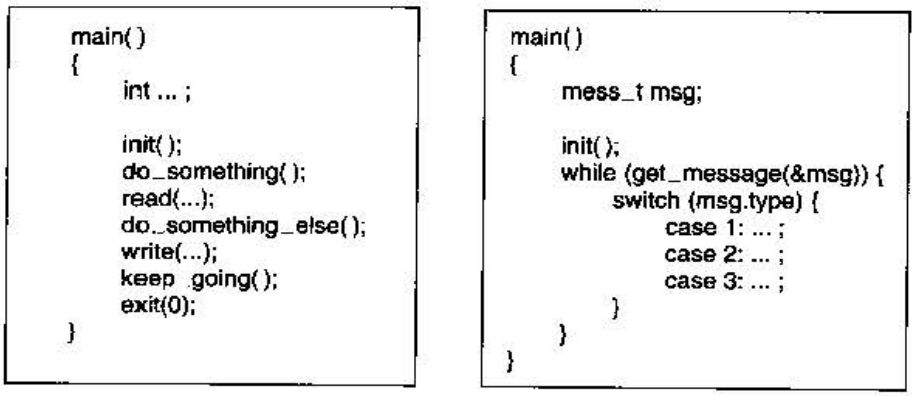

图13-1 a) 算法代码；b) 事件驱动代码

另一种执行范型是图13-1b所示的事件驱动范型（event-driven paradigm）。在这里程序执行某种初始化（例如通过显示某个屏幕），然后等待操作系统告诉它第一个事件。事件经常是键盘敲击或鼠标移动。这一设计对于高度交互式的程序是十分有益的。

这些做事情的每一种方法造就了其特有的程序设计风格。在算法范型中，算法位居中心而操作系统被看作是服务提供者。在事件驱动范型中，操作系统同样提供服务，但是这一角色与作为用户行为的协调者和被进程处理的事件的生产者相比就没那么重要了。

3. 数据范型

执行范型并不是操作系统导出的惟一范型，同等重要的范型是数据范型。这里关键的问题是系统结构和设备如何展现给程序员。在早期的FORTRAN批处理系统中，所有一切都是作为连续的磁带而建立模型。用于读入的卡片组被看作输入磁带，用于穿孔的卡片组被看作输出磁带，并且打印机输出被看作输出磁带。磁盘文件也被看作磁带。对一个文件的随机访问是可能的，只要将磁带倒带到对应的文件并且再次读取就可以了。

使用作业控制卡片可以这样来实现映射：

```
MOUNT(TAPE08, REEL781)
RUN(INPUT, MYDATA, OUTPUT, PUNCH, TAPE08)
```

第一张卡片指示操作员去从磁带架上取得磁带卷781，并且将其安装在磁带驱动器8上。第二张卡片指示操作系统运行刚刚编译的FORTRAN程序，映射INPUT（意指卡片阅读机）到逻辑磁带1，映射磁盘文件MYDATA到逻辑磁带2，映射打印机（称为OUTPUT）到逻辑磁带3，映射卡片穿孔机（称为PUNCH）到逻辑磁带4，并且映射物理磁带驱动器8到逻辑磁带5。

FORTRAN具有读写逻辑磁带的语法。通过读逻辑磁带1，程序获得卡片输入。通过写逻辑磁带3，输出随后将会出现在打印机上。通过读逻辑磁带5，磁带卷781将被读入，如此等等。注意，磁带概念只是集成卡片阅读机、打印机、穿孔机、磁盘文件以及磁带的一个范型。在这个例子中，只有逻辑磁带5是一个物理磁带，其余的都是普通的（假脱机）磁盘文件。这只是一个原始的范型，但它却是正确方向上的一个开端。

后来，UNIX问世了，它采用”所有一切都是文件”的模型进一步发展了这一思想。使用这一范型，所有I/O设备都被看作是文件，并且可以像普通文件一样打开和操作。C语句

```
fd1 = open("file1", O_RDWR);
fd2 = open("/dev/tty", O_RDWR);
```

打开一个真正的磁盘文件和用户终端。随后的语句可以使用fd1和fd2分别读写它们。从这一时刻起，在访问文件和访问终端之间并不存在差异，只不过在终端上寻道是不允许的。

UNIX不但统一了文件和I/O设备，它还允许像访问文件一样通过管道访问其他进程。此外，当支持映射文件时，一个进程可以得到其自身的虚拟内存，就像它是一个文件一样。最后，在支持/proc文件系统的UNIX版本中，C语句