第4章 概 述

本章的目的是对源代码作为整体进行综述，亦即先看森林，后看树木。

查看源代码将发现它由44个文件组成，其中：

• 两个是汇编语言文件，它们的名字带后缀“ .s ”；

• 28个是用“ C ”语言编写的，它们的名字带后缀“ .c ”；

• 14个是用“ C ”语言编写的，但并不准备单独编译，名字带后缀“ .h ”。

这些文件以及它们的内容是由编写它们的程序设计人员为使他们自己使用方便而安排的，并没有考虑读者阅读的便利。在很多方面，文件之间的划分与本书的讨论无关。

如同在第1章中已提及的，所有这些文件已被组织成五个部分。在组织这些部分时所遵循的基本点是：尽可能使各部分的长度大致相近；将强相关的文件分在同一部分中；将弱相关的文件分到不同部分中。

4.1 变量分配

PDP11系统结构支持对变量进行高效存取，其条件是这些变量的绝对地址已知，或者它们相对于栈指针的地址可在编译时完全确定。

PDP11在硬件方面并无对变量说明的多词法级的支持，在块结构语言，例如 Algol或Pascal，则有变量说明的多词法级支持。由于“ C ”在PDP11上实现，所以它只支持两个词法级：全局(global)和局部(local)。

对全局变量进行静态分配；对局部变量则在当前栈中或在通用寄存器 (r2、r3和r4用于此种方式)中进行动态分配。

4.2 全局变量

除极个别例子外，对全局变量的说明都已集中到各个“ .h ”文件中。例外包括：

1) 在“ swtch ”中说明的静态变量“ p ”(2180)，它以全局方式存储，但只能在“ swtch ”过程中存取。(在UNIX中，“ p ”是一个普通使用的局部变量名。)

2) 有些变量，例如“ swbuf(4721)，仅由同一文件中的若干过程引用，于是在该文件的开始部分对它们进行说明。

可在引用全局变量的每个文件中分别对它们进行说明。然后，装入程序 (loader)在将编译好的各程序文件连接到一起时，将每一变量的各个说明相匹配。

4.3 “ C ”预处理程序

如果全局性说明在每个文件必须全部重复 (正如在Fortran中所要求的那样)，那么程序量会

显著增加，而且对一个说明进行修改是非常枯燥无味的，同时非常容易出错。

UNIX使用了“ C ”编译的预处理程序设施，从而避免了这些困难。这种设施允许将大多数全局变量的说明记录在少数几个“ .h ”文件之中，每个全局变量记录一次。

任何时候若需一特定全局变量的说明，则只要将相应“ .h ”文件包含(“ include ”)在将被编译的文件中。

UNIX也使用“ .h ”文件作为载体存放标准符号名的列表以及某些结构类型的说明。符号名常代表常数和可调整的参数。

例如，若文件“ bottle.c ”包含一过程“ glug ”，而该过程引用一名为“ gin ”的全局变量，该全局变量在文件“ box.h ”中说明，那么下列语句应插入到文件“ bottle.c ”的开始处：

```
#include“box.h”
```

当编译该文件时，所有在“ box.h ”中的说明都被编译，因为它们是在“ bottle.c ”中的任一过程之前发现的，所以在所产生的可重定位模块中它们被标记为外部的。

当将所有目标模块连接到一起时，在其源文件中包括了“ box.h ”的每个文件中都将发现对“ gin ”的引用。所有这些引用将是一致的，装入程序将为“ gin ”分配一个所需的地址空间，并相应调整所有对“ gin ”的引用。

4.4 第一部分

第一部分包含了很多“ .h ”文件和汇编语言文件。它也包含若干涉及系统初启和进程管理的文件。

4.4.1 第1组“ .h ”文件

param.h，不包含变量说明，但包含很多操作系统常数和参数的定义，以及三个简单结构的说明。有关约定是对于定义的常数只使用大写字母。

systm.h〔第19章〕，主要由说明组成，其中作为副作用定义了“ callout ”和“ mount ”结构。注意，其中没有一个变量是显式地赋予初值的，它们的初值都赋为 0。

头三个数组的长度是定义在“ param.h ”中的参数。因此，任一包含“ systm.h ”的文件，必须在包含“ systm.h ”之前先包含“ param.h ”。

seg.h，包含几个定义和一个说明，在引用段寄存器时使用这些定义和说明。此文件实际上可并入“ param.h ”和“ systm.h ”中。

proc.h〔第7章〕，包含重要的“ proc ”的说明。proc是一个结构类型，也是这种结构类型的一个数组。proc结构的每一个元素的名字都以“ p- ”开始，没有其他变量是这样命名的。类似的惯例用于命名其他结构的各元素。

proc结构中头两个元素“ p_stat ”和“ p_flag ”的各种相关值都有单独的名字，它们也在“ proc.h ”中定义。

user.h〔第7章〕，包含非常重要的“ user ”结构的说明，还包含一组与“ u_error ”相关的定义值。在任一时刻只有一个“ user ”结构的实例是可存取的。该实例以名字“ u ”引用，它

位于长度为1024字节的“每个进程数据区”的低地址部分。

一般而言，本书在此后不会对“.h”文件进行详细而全面的分析。希望读者能反复地参阅它们，这样就会不断增加对它们的熟悉和理解程度。

4.4.2 汇编语言文件

有两个汇编语言文件，大约占源代码中的10%。应当对这些文件有较高的熟悉程度。

low.s〔第9章〕，包含初始化内存低地址部分的有关信息，包括陷入矢量。此文件是由名为“mkconf”的公用程序生成的，以适应在特定配置中具有的外部设备集。

m40.s〔第6、8、9、10、22章〕，包含一组与PDP11/40相适应的例程，它们执行大量“C”不能直接实现的特殊函数。我们将在适当的时间和位置对此文件的各部分进行介绍和讨论。(最大的汇编语言过程“backup”已作为一个练习留给读者分析。)

当使用PDP11/45或PDP11/70时，应将“m40.s”代换成“m45.s”但在这里我们没有提供此文件。

4.4.3 在第一部分中的其他文件

main.c〔第6、7章〕，包含“main”，它执行多种初始化任务以使UNIX运行。它也包含“sureg”和“estabur”，它们设置用户态段寄存器。

slp.c〔第6、7、8、14章〕，包含进程管理所需的各主要过程，包括：“newproc”、“sched”、“sleep”和“swtch”。

prf.c〔第5章〕，包含“panic”和若干其他过程，它们
显示初始化消息和出错消息。

malloc.c〔第5章〕，包含“malloc”和“mfree”，它们

4.5 第二部分

第二部分涉及陷入(traps)、硬件中断(hardware interrup

陷入和硬件中断造成CPU正常指令执行序列的突然切
外产生的特殊条件的机制。

这种设施也用作“系统调用”机制的一部分，在“系
执行一条“trap”指令以故意造成一次陷入，从而取得操作

软件中断(或称“信号”)是一种特殊的进程间进行通信

reg.h〔第10章〕，定义了一组常数，当前用户态寄存器
它们时，使用这一组常数。

trap.c〔第12章〕，包含“C”过程“trap”，其功能是识

sysent.c〔第12章〕，包含数组“sysent”的说明及初值
类型找到在核心态下执行的相应例程的入口地址。

sys1.c〔第12、13章〕，包含多个与系统调用相关的例

和“ fork ”。

sys4.c〔第12、13、19章〕，包含“ unlink ”、“ kill ”及若干其他非主要系统调用的例程。

clock.c〔第11章〕，包含“ clock ”例程，其主要功能是处理时钟中断，并且进行偶发事务及会计事务的大量处理。

sig.c〔第13章〕，包含处理“ 信号 ”或“ 软件中断 ”的过程。这些为进程通信及跟踪提供了能力。

4.6 第三部分

第三部分涉及内存(主存)和磁盘存储器之间的基本输入/输出操作。这些操作对程序换进/换出以及磁盘文件的创建和引用活动是基础性的。

本部分也包含使用和处理大缓存(512字节)的各个过程。

text.h〔第14章〕，定义“ text ”结构和数组。系统使用一个“ text ”结构定义一共享正文段(shared text segment)的状态。

text.c〔第14章〕，包含管理共享段的各个过程。

buf.h〔第15章〕，定义“ buf ”结构和数组、“ devtab ”结构以及“ b-error ”可包含的各种值的符号名。为管理大缓存，所有这些都是必须的。

conf.h〔第15章〕，定义结构数组“ bdevsw ”和“ cdevsw ”，它们指定了面向设备的各处理过程，为执行针对设备的逻辑文件操作，需调用这些过程。

conf.c〔第15章〕，与“ low.s ”相似，“ conf.c ”是由公用程序“ mkconf ”生成的，以适应特定系统配置的各种外部设备。它包含“ bdevsw ”和“ cdevsw ”数组的初始化值表，这两个

I/O操作。

m40.s ”之后的最大一个文件，它包含了处理大缓存以及

K05磁盘控制器的设备驱动程序。

。

，以及目录组成，所有这些都组织存放在单个存储设备，

”有关的代码。

结构和数组

sys ”结构，在装配和拆卸文件系统时，从超级块 (super

ino.h，说明记录在“已装配”设备上的“索引节点”(“inodes”)结构。此文件并不被任何其他文件包括，仅仅是为提供信息而存在。

inode.h〔第18章〕，定义“inode”结构和数组。在管理进程对文件的存取时，“inode”起关键性作用。

sys2.c〔第18、19章〕，包含一组与系统调用相关的例程，这些系统调用是“ read”、“write”、“creat”、“open”和“close”。

sys3.c〔第19、20章〕，包含一组与多个非主要系统调用相关的例程。

rdwri.c〔第18章〕，包含与读/写文件有关的中间层次例程。

subr.c〔第18章〕，包含与i/o有关的中间层次例程，特别是“ bmap”，它将逻辑文件指针翻译成物理磁盘地址。

fio.c〔第18、19章〕，包含与文件打开、关闭和存取控制有关的中间层次例程。

alloc.c〔第20章〕，包含若干管理文件系统 inode和磁盘块资源的过程，它们负责分配“inode”数组表目项和磁盘存储块。

iget.c〔第18、19、20章〕，包含与更新和引用“inode”有关的各过程。

nami.c〔第19章〕，包含“namei”过程，它搜索文件目录。

pipe.c〔第21章〕，是“pipe”(管道)的“设备驱动程序”，“pipe”是特殊形式的短磁盘文件，用于从一个进程向另一个进程传输信息。

4.8 第五部分

第五部分是最后一部分。它与低速面向字符的外部设备的输入 /输出相关。

这类设备共享一个公共缓冲池，该缓冲池由一组标准的过程进行管理和操作。

面向字符的外设用下列设备作为实例：

KL/DL11 交互式终端。

PC11 纸带阅读/穿孔机。

LP11 行式打印机。

tty.h〔第23、24章〕，定义“clist”结构(用作字符缓冲队列的队首)和“tty”结构(存放控制某单个终端的相关数据)，说明“partab”表(用于控制单个字符向终端的传输)，也定义很多相关参数的符号名字。

kl.c〔第24、25章〕，是经KL11或DL11接口相连终端的设备驱动程序。

tty.c〔第23、24、25章〕，包含若干独立于接口的公共过程，它们控制向 /从终端的传输，也考虑到各种终端的特殊性。

pc.c〔第22章〕，是PC11纸带阅读/穿孔控制器的设备处理程序。

mem.c，包含一组存取主存的过程，它们将主存视作为普通文件。这部分代码作为一个练习留给读者进行分析。