### 符号分为全局符号（由本模块定义并能被其他模块引用的符号）、外部符号（有其他模块定义并被本模块引用的全局符号）和局部符号（带static属性的C函数和全局变量）

1. 有三个特殊的伪节，ABS表示不该被重定位的符号，UNDEF表示未被定义的符号，也就是在本模块中被引用但却在其他地方定义的符号，COMMON表示还未被分配位置的未初始化的数据目标。
2. 在Linux系统中，静态库以一种被称为存档的特殊文件格式存放在磁盘中，存档文件是一组链接起来的可重定位目标文件的集合。
3. 可执行文件连续的片被映射到连续的内存段
4. 在x86-64系统中，代码段总是从虚拟地址0x400000处开始，后面是数据段，运行时堆在数据段之后，堆后面的区域为共享模块保留，用户栈总是从最大的合法用户地址开始，向较小内存地址增长
5. 由于.data段有对齐要求，所以代码段和数据段之间是有间隙的，共享库是一个共享模块，在运行或加载时可以加载到任意的内存地址并和一个在内存中的程序链接起来，这个过程称为动态链接。
6. 在任何给定的文件系统中，对于一个库只有一个.so文件，所有引用该库的可执行目标文件共享这个.so文件中的代码和数据，在内存中，一个共享库的.text节的一个副本可以被不同的正在运行的进程共享。动态链接器本身就是一个共享目标，如ld-linux.so，加载器会加载和运行这个动态链接器。
7. 可以加载而无需重定位的代码称为位置无关代码PIC，可以用-fpic选项生成PIC代码，共享库的编译必须总是使用该选项。
8. 无论我们在内存中的何处加载一个目标模块，数据段与代码段的距离总是保持不变，代码段中任何指令和数据段中的任何变量之间的距离都是一个运行时常量，与代码段和数据段的绝对内存位置是无关的。
9. GOT是数据段的一部分，GOT是一个数组，每个条目是8字节地址，和PLT联合使用时，GOT[0]和GOT[1]包含动态链接器在解析函数地址时会使用的信息，GOT[2]是动态链接器在ld-linux.so中的入口点，其余每个条目对应于一个被调用的函数，每个条目都有一个向匹配的PLT条目，初始时每个GOT条目都指向PLT条目的第二条指令；而PLT是代码段的一部分，PLT是一个数组，每个条目是16字节代码，PLT[0]跳转到动态链接器中，PLT[1]调用系统启动函数初始化执行环境，调用main函数并处理其返回值，PLT[2]开始的条目调用用户代码调用的函数
10. 在处理异常时x86-64会把包含当前条件码的EFLAGS寄存器和其他内容压入栈中，如果控制从用户模式转移到内核，那么所有这些项目都被压到内核栈中
11. 异常处理程序运行在内核模式下
12. 在系统调用中，寄存器%rax会被破坏，%rdi,%rsi,%rdx,%r10,%r8,%9包含至多6个参数，寄存器%rcx,%r11会被破坏，%rax包含返回值，负的返回值表示发生了错误，对应于负的errno
13. 处理器用某个控制寄存器中的一个模式位来提供用户模式和内核模式的功能，设置了模式位表示内核模式，反之表示用户模式，/proc文件系统允许用户模式进程访问内核数据结构的内容
14. Linux系统级函数如果出错通常会返回-1，而pause函数总是返回-1
15. 进程收到SIGSTOP,SIGTSTP,SIGTTIN,SIGTTOU信号可以使进程停止直到其收到一个SIGCONT信号继续执行
16. 内核会安排init进程作为任何孤儿进程的养父，init进程的pid为1，是在系统启动时由内核创建的，不会终止，是所有进程的祖先。
17. shell中的最后一个参数如果是&表示在后台执行程序
18. SIGKILL和SIGSTOP既不能被捕获也不能被忽略
19. 键入ctrl C会向前台进程组发送一个SIGINT，键入ctrl Z会向前台进程组发送一个SIGTSTP信号挂起前台进程
20. 在发送信号时，若pid>0表示发信号给进程pid，若pid<0表示发信号给进程组-pid
21. alarm函数可以在sec秒后给自己发送一个SIGALRM信号，返回值为前一次闹钟剩余的秒数（若没有返回0），且任何新的alarm的设置都会取消先前等待的alarm，
22. 调用信号处理程序称为捕获信号，执行信号处理程序称为处理信号
23. 内核默认阻塞任何当前处理程序正在处理信号类型的待处理的信号
24. 安全的信号处理程序的准则：处理程序要尽可能简单、只调用异步信号安全的函数（要么是可重入的，要么不能被信号处理程序中断）（printf,sprintf,malloc,exit均不安全），可以使用sio包来打印简单的信息、保存和恢复errno（只在处理程序要返回时才有此必要）、阻塞所有的信号，保护对共享全局数据结构的访问、用volatile声明全局变量、用sig\_atomic\_t声明标志保证读写的原子性。
25. 不可以用信号来对其它进程中发生的事件计数
26. Signal包装函数设置了信号处理程序，保证只要可能会自动重启被中断的系统调用，并且会一直保持设置的信号处理程序直到新的调用
27. setjmp的返回值不能被赋值给变量，但可以安全地用在switch或条件语句的测试中
28. 信号处理程序可以使用sigsetjmp和siglongjmp，但sigsetjmp和siglongjmp不是异步信号安全的，原因是siglongjmp可以跳到任意的代码，所以必须在siglongjmp可达的代码只中调用安全的函数
29. 初次调用setjmp时其返回值为0，从longjmp返回时其返回值非0
30. DRAM缓存不命中比SRAM缓存不命中要昂贵很多，从磁盘的一个扇区读取第一个字节的时间开销要比读这个扇区中连续的字节慢很多，因此虚拟页往往很大，同时DRAM缓存是全相联的，DRAM缓存通常使用写回而不是直写。
31. 等到有不命中发生时才换入页面的策略称为按需页面调度，所有现代系统使用的都是按需页面调度。
32. 局部性原则保证在任意时刻，程序倾向于在一个较小的活动页面集合上工作，这个集合叫工作集或常驻集合，但如果工作集大小超过了物理内存大小，那么程序将会出现抖动。
33. 加载器从不从磁盘到内存实际复制任何数据，而是在每个页初次被引用时进行按需页面调度。
34. 在进行malloc时操作系统会分配k个连续的虚拟内存页面，并将他们映射到物理内存中任意k个不一定连续的位置
35. TLB是虚拟寻址的缓存，而大部分系统中SRAM高速缓存都进行物理寻址
36. Core i7支持48位虚拟地址空间和52位物理地址空间，Corei7处理器封装包括4个核，所有核共享一个L3cache和一个DDR3内存控制器，每个核包含一个层次结构的TLB，一个层次结构的i-cache和d-cache，一组快速的基于quickpath的点到点链路，这种链路可以让一个核与其他核的外部I/O桥直接通信。TLB虚拟寻址，四路组相连，L1,L2,L3cache物理寻址，块大小64字节，L1,L2八路组相连，L316路组相连，Linux使用4K的页。
37. 一个Linux进程在运行时，与已分配了的页相关联的页表都是驻留在内存中的，物理页表和物理页都是4KB对齐的。
38. Core i7在PTE项中有三个权限位控制对页的访问：R/W限制只读和读写；U/S确定是否能在用户模式下访问；XD（禁止执行）是64位系统中引入的，降低缓冲区溢出攻击的风险；内核缺页处理程序需要两个位：A位表示引用位，每次访问由MMU设置，由软件清除，可以用来实现页面替换算法；D位为脏位或修改位，告诉内核在替换页之前是否需要写回牺牲页，内核可以通过一条特殊的内核模式的指令来清楚引用和修改位。
39. Linux会将一组连续的虚拟页面映射到相应的一组连续的物理页面，这就为内核提供了一种便利的方法来访问物理内存中任何特定的位置。Linux将虚拟内存组织成一些区域（段）的集合，一个区域就是已经存在的虚拟内存的连续片，这些页以某种形式相关联，每个存在的虚拟页都保存在某个区域中，内核为每个进程维护一个单独的数据结构task\_struct，其中一个条目指向mm\_struct，其中有两个字段pgd和mmap，pgd指向第一级页表的基地址，而mmap指向一个mm\_area\_structs的链表
40. 虚拟内存区域可以映射到两种类型的对象中的一种，一个区域可以映射到一个普通磁盘文件的连续部分，区域比文件区要大就用0来填充；一个区域也可以映射到一个匿名文件，这样的文件由内核创建，包含的全是二进制0，在第一次引用时内核在物理内存中找到一个合适的牺牲页面，用二进制0覆盖牺牲页面并更新页表，在磁盘和内存之间没有数据传送，这样的文件称为请求二进制0的。一旦一个虚拟页面被初始化了，它就在一个由内核维护的交换文件之间换来换去。
41. 一个对象可以被映射到虚拟内存的一个区域，如果作为共享对象，那么这个进程对这个区域的任何写操作对也映射这个共享对象的其他进程也可见，且会反映到原始文件中。而映射到私有对象的改变对其他进程是不可见的，也不会反映到磁盘上的文件，映射到共享对象的区域称为共享区域，反之称为私有区域。私有对象使用写时复制（区域结构中的标记）技术映射到内存中。
42. 创建新进程时内核会将两个进程中的每个页面都标记为只读，并将每个区域结构都标记为私有的写时复制。加载并运行可执行文件需要删除当前进程虚拟地址空间已存在的用户区域、映射私有区域、bss区域是请求二进制0的映射到匿名文件；栈和堆区域也是请求二进制0的，初始长度为0；映射共享区域最后设置程序计数器。
43. 在32位模式中，malloc返回的块的地址总是8的倍数，而在64位模式中该地址总是16的倍数；用一个负数调用sbrk是合法的；如果free一个不是已分配块的起始位置的指针那么其行为是未定义的，free函数什么都不返回，就不会告诉应用出现了错误。
44. 显式分配器要求能处理任意请求序列，不能假设分配和释放请求的顺序，同时必须立刻响应请求，只能使用堆，要对齐块，不能修改已经分配的块。
45. 内部碎片是一个已分配块比有效载荷大的时候发生的，其大小就是已分配块的大小和它们有效载荷大小之差的和。
46. 一个块的头部编码了这个块的大小，包括头部和所有填充，头部后面就是请求的有效载荷，系统对齐要求会对分配器上的最小块大小有强制的要求，比如如果要求8字节对齐，那么所有块大小都必须是8的倍数。
47. 放置策略常见的有首次适配（从头搜索空闲块，选择第一个合适的空闲块）、下一次适配（从上一次查询结束的地方开始适配）和最佳适配（选择大小最合适的块），首次适配会在开头留下一些小空闲块的碎片，这样增加了对大空闲块的访问时间；而下一次适配比首次适配更快，但下一次适配的内存利用率更低；最佳适配内存利用率最高，但在隐式空闲链表中所需时间最长。
48. 合并策略包括立即合并和推迟合并，立即合并可以在常数时间内完成，但在一些抖动的情况中会出现一个块被合并后马上分隔的情况，而在一些快速分配器中会使用某种形式的推迟合并。
49. 可以把前一个块是否空闲的标志存放在本块的低位里，这样已分配块就不需要footer了，但未分配块还需要footer
50. 教材上的隐式空闲链表的实现开头有一个4byte的填充块，接下来是一个8byte的已分配的序言块，其有header和footer且永不释放，分配器的heap\_listp总是指向序言块，堆以一个大小为0的只有一个header的结尾块结束，这个块标志为已分配
51. 显式空闲链表：使用LIFO策略维护链表将新释放的块放在链表的开头，释放一个块可以在常数时间，如果使用footer那么合并也可以在常数时间内完成；也可以按照地址顺序维护链表，按地址顺序维护的内存利用率更高，显式链表的缺点是空闲块要足够大能放下需要的指针、头部和脚部，这潜在提高了内部碎片的程度。
52. 简单分离存储每个大小类的空闲链表包含大小相等的块，每个块的大小就是这个类的最大大小，我们不会分割空闲块，这样分配和释放块都是很快的常数时间操作，且每个块只有很小的内存开销，但很容易造成内部碎片和外部碎片；而分离适配维护一个空闲链表的数组，每个空闲链表对应一个大小类，对分离空闲链表的首次适配搜索其空间利用率接近于最佳适配
53. 伙伴系统每个大小类都是2的幂次，为每个2的幂次块大小维护一个分离空闲链表，请求块向上舍入到最接近的2的幂次，每次分配一个块都将一个合适的空闲块二分，分下来的空闲块放在相应的链表中，一个块的地址和其伙伴只有一位不相同，这样可以实现快速搜索和快速合并，但缺点是严重的内部碎片。
54. 垃圾收集是自动回收堆存储的过程，在支持垃圾手机的系统中应用显式分配块但从不显式释放它们。垃圾收集器将内存视为一张有向可达图，该图中的节点被分成一组根节点和一组堆节点，每个堆节点对应于对重一个已分配块，有向边表示一个节点中的某个位置指向另一个节点中的某个位置，根节点对应于不在堆中但指向堆中的指针，这些位置可以是寄存器、栈里的变量或虚拟内存中读写数据区中的全局变量。如果存在一条从任意根节点到某个节点的路径那么该节点就是可达的，否则该节点就是不可达的，不可达节点对应于垃圾。C和C++的垃圾收集器不能维持可达图的精确表示，是保守的垃圾收集器。（每个可达节点都被标记为可达，但不可达节点不一定）
55. Mark&Sweep垃圾收集器有mark阶段和sweep阶段组成，标记阶段标记根节点的所有可达和已分配的后继，清除阶段释放每个未标记的已分配块。但C程序为isptr函数的实现造成了挑战：没有明显的方式判断其输入参数是否是一个指针，也没有明显的方式来判断p是否指向一个已分配块的有效载荷中的某一个位置——将已分配块的集合维护成一颗平衡二叉搜索树，这样执行对这棵树的二分查找依赖于块头部的大小字段判断p是否在这个块的范围内，这样能保证标记所有根节点可达的节点，但并不能保证释放所有空闲块，因此是保守的，但C语言的垃圾收集必须是保守的，因为C语言中的标量可以伪装成指针。
56. 堆内存并不总是初始化为0；一元运算符--和\*的优先级相同，因此会从右向左结合；如果对局部变量使用&，那么会得到其在栈上的地址，在另一个函数中使用这个地址会错误地修改栈的内容；也不能引用空闲堆块的数据。
57. 目录是包含一组链接的文件，每个链接将一个文件名映射到一个文件，每个目录至少含有两个条目：.是到该目录自身的链接，而..是到目录层次结构中父目录的链接，可以用mkdir创建一个目录，用ls查看其内容，用rmdir删除该目录，名为/的目录是根目录，每个进程都有一个当前工作目录表示其在目录层次结构中的当前位置作为其上下文的一部分
58. 绝对路径名以一个/开始表示从根节点开始的路径；相对路径名以文件名开始表示从当前工作目录开始的路径
59. open返回的描述符是当前没有打开的最小描述符，O\_TRUNC会把已存在的文件清空，而O\_CREAT在文件不存在时会创建空文件但不会把已存在的文件清空
60. open的第三个参数指定了新文件的访问权限位，每个进程有一个umask，这存在于这个进程的上下文中，而第三个参数设置一个mode，那么最终的文件的权限就是mode&(~umask)
61. read函数返回的是ssize\_t而不是size\_t，因为read在出错时需要返回-1，ssize\_t对应于long
62. 带缓冲区的rio函数是线程安全的，在同一描述符上可以被交错地调用
63. rio无缓冲区函数：rio\_readn和rio\_writen，可以在内存和文件之间直接传送数据，rio\_readn在遇到EOF时会出现不足值，而rio\_writen不会遇到不足值，如果他们被一个从应用信号处理程序的返回中断，那么每个函数会手动地重启read或write
64. rio有缓冲区函数：每次打开一个描述符，都会调用rio\_readinitb函数将描述符fd与地址rp处的一个类型为rio\_t的缓冲区联系起来，对只含文本行的文件，rio\_readlineb可以从内部读缓冲区中复制一个文本行，当缓冲区变空时会调用read函数填满缓冲区，而对二进制文件则可以使用rio\_readnb（rio\_readn的有缓冲区的版本），rio\_readnb和rio\_readlineb的调用可以交替进行，但不应和rio\_readn交替使用
65. 应用程序可以使用stat和fstat函数检索关于文件的信息（文件的元数据），stat以文件名为输入，而fstat以文件描述符为输入
66. 应用程序可以用readdir来读目录的内容，opendir以路径名为参数返回指向目录流的指针，用closedir来关闭流并释放所有资源
67. 每个进程有独立的描述符表，所有进程共享文件表和v-node表
68. 在dup2中，如果newfd已经打开了，那么在覆盖前会先将其关闭，不要求newfd必须打开
69. 标准I/O库将一个打开的文件模型化为一个流，一个流就是一个指向File\*类型的结构的指针，类型为File的流是对文件描述符和流缓冲区的抽象。
70. I/O的使用方式：尽可能使用标准I/O；不要使用scanf或rio\_readlineb来读二进制文件；对网络套接字的I/O使用rio函数
71. 标准I/O流从某种意义上讲是全双工的，但如果中间没有插入对fflush、fseek、fsetpos或rewind的调用，一个输入函数不能跟随在一个输出函数之后，如果中间没有插入fseek、fsetpos或rewind的调用一个输出函数不能跟随在一个输入函数之后，而在网络编程中是禁止使用lseek的，因此建议在网络套接字上使用rio函数进行输入输出
72. 客户端和服务器是进程，而不是机器或主机
73. 网络是一个按照地理远近组成的层次系统，最底层是LAN局域网，最流行的局域网技术是以太网，一个以太网段包含一些电缆和一个集线器，电缆一端连接主机的适配器，另一端连在集线器的端口上，集线器不加分辨地把每个端口收到的每个位复制到另一个端口上，每个以太网适配器都有全球唯一的48位地址，存储在非易失性存储器上。
74. 在局域网内数据组织成帧进行传递，每个帧包括一些头部位和有效载荷。
75. 使用电缆和网桥，多个以太网段可以连接成较大的局域网，形成桥接以太网，桥接以太网中的电缆的带宽可以是不同的，网桥只在有必要时有选择地将帧从一个端口复制到另一个端口从而节约其他网段上的带宽
76. 多个不兼容的局域网可以通过路由器连接起来，每台路由器对他所连接到的每个网络都有一个适配器（端口），路由器也能告诉点到点电话连接，这称为WAN广域网
77. 协议软件必须提供命名机制和传送机制。命名机制：每台主机会被分配至少一个互联网络地址，这个地址唯一标识了这台主机；传送机制：互联网络协议把数据位捆扎成包，一个包由包头和有效载荷构成
78. 主机和路由器在不兼容的局域网间传送数据的方式是先把数据附加一个互联网络包头和LAN1帧头，通过LAN1适配器达到协议软件后协议软件根据包头剥落LAN1帧头，附加LAN2包头传给LAN2，主机B上的协议软件剥落包头和LAN2帧头获得数据，这样每个帧的有效载荷是一个互联网络包，而一个包的有效载荷才是数据。
79. 网络的分层模型：

|  |
| --- |
| 应用层：  HTTP（基于文本的应用级协议，超文本传输协议）  DNS（域名解析，域名系统数据库，由上百万条主机条目结构组成，每一条定义了一组域名和一组IP地址之间的映射）  FTP（文件传输协议）  SMTP（简单邮件传输协议）  MIME（多用途的网际邮件扩充协议）  CGI（通用网关接口） |
| 传输层：  TCP（可靠的面向连接的传输控制协议，定义了大端字节顺序）  UDP（用户数据报协议，无连接，不可靠，尽力而为的服务，包可以在进程间传递） |
| 网络层：  IP（网际协议，提供基本的命名方法和递送机制，发送数据报，但不可靠，因为不会试图恢复重复或丢失的数据报） |
| 数据链路层（网桥，交换机） |
| 物理层（中继器、集线器） |

1. 因特网可以看做一个世界范围内的主机集合，主机集合被映射为一组32位的地址，这组IP地址被映射为一组因特网域名的标识符，因特网主机上的进程可以通过连接和其他任何因特网主机上的进程通信。
2. IPV4（AF\_INET）使用32位地址，而IPV6（AF\_INET6）使用128位地址
3. 没有处理64位值的主机字节与网络字节的转换函数
4. 域名集合形成了一个层次结构，可以表示成一棵树，树的节点表示域名，反向到根（叶节点——根节点）的路径形成了域名，第一层是未命名的根节点，下一层是一级域名，由ICANN定义，如.com,.edu,.net等，而二级域名由ICANN的各个授权代理按先到先得的顺序分配，一个组织得到二级域名后可以在这个子域中创建任何新的域名
5. 可以用hostname确定本机实际域名，localhost总是映射为回送地址127.0.0.1
6. 一个套接字是连接的一个端口，每个套接字都有相应的套接字地址，是由一个因特网地址和一个16位整数端口组成的（与硬件端口没有关系）
7. 客户端套接字地址的端口由内核自动分配，称为临时端口，而服务器套接字端口通常是某个知名端口，与这个服务相对应，比如Web服务器用端口80，电子邮件服务器使用端口25，每个具有知名端口的服务器都有一个对应的知名的服务名，比如Web服务器的名字是http，而email的知名名字是smtp
8. 套接字接口是一组函数
9. 从Linux内核角度看，一个套接字就是通信的一个端点；从Linux程序的角度看，一个套接字就是有相应描述符的打开文件
10. IP地址和端口号总是以大端法存放的
11. socket函数可以创建一个套接字描述符，客户端可以调用connect函数建立和服务器的连接，bind函数会将服务器套接字地址和套接字描述符联系起来，listen函数将一个套接字从一个主动套接字转化为一个监听套接字，该套接字可以接受来自客户端的连接请求，连接请求是可以排队的，因此如果connect失败，客户端会被阻塞在rio\_readlineb上，最后accept函数等待来自客户端的连接到达监听描述符，然后返回一个已连接描述符进行通信
12. getaddrinfo将主机名，主机地址、服务名和端口号的字符串表示转化为套接字地址结构，这个函数是可重入的并适用于任何协议，返回一个指向addrinfo结构的链表，每个结构指向一个对应于host和service的套接字地址结构，客户端调用该函数后会遍历这个链表，直到socket和connect成功，而服务器则会遍历这个链表，直到socket和bind成功。程序最后必须使用freeaddrinfo释放。getaddrinfo的参数host可以是域名，也可以是数字地址，而service可以是服务名，也可以是十进制端口号，host和service都可以设置为NULL，但必须指定二者中至少一个
13. getaddrinfo对每个地址默认最多返回三个addrinfo结构，将ai\_socktype字段设置成SOCK\_STREAM可以只返回一个，这是我们希望的行为
14. getnameinfo与getaddrinfo相反，将一个套接字地址结构转换成相应地主机和服务名字符串，是可重入的和协议无关的
15. 客户端可以调用高级封装函数open\_clientfd来建立和服务器的连接，其有两个参数hostname和port，表示服务器运行在hostname，在port端口上监听连接请求，我们会遍历得到的链表并尝试socket和connect直到成功，如果失败则要小心关闭套接字描述符
16. 服务器可以用open\_listenfd创建一个监听描述符并准备好接收连接请求。在该函数中使用了setsockopt函数使得服务器能够被终止、重启和立刻开始接受连接请求，一个重启的服务器默认将在大约30s内拒绝客户端的连接请求。在调用getaddrinfo时使用AI\_PASSIVE标志并将host设为NULL得到套接字地址结构中的地址字段是通配符地址，告诉内核这个服务器会接受发送到该主机所有IP地址的请求，这是我们希望的。最后调用listen函数，如果listen失败同样要小心地关闭描述符。
17. 对于因特网链接，当一个进程关闭连接它的那一段时会发生EOF，另一端的进程试图读取流中最后一个字节之后的文件时会检测到EOF
18. Web和常规的文件检索服务（如FTP）的主要区别是Web内容可以用HTML语言编写，其使得一个页面可以包含指针（超链接），这些指针可以指向存放在任何因特网主页上的内容。
19. Web服务器以两种不同的方式向客户端提供内容：取一个磁盘文件并将其内容返回给客户端，磁盘文件称为静态内容，而返回文件的过程称为服务静态内容；运行一个可执行文件并将其输出返回给客户端，运行时可执行文件产生的输出称为动态内容，而运行程序并返回其输出的过程称为服务动态内容
20. 每条由Web服务器返回的内容都是和他管理的某个文件相关联的，这些文件中的每一个都有一个唯一的名字，叫做URL（通用资源定位符）。一个可执行文件的URL可以在文件名后包括程序参数，？分隔文件名和参数，每个参数用&分隔开，URL后缀中的/不表示根目录，而表示被请求内容类型的主目录，最小的URL后缀是/，所有服务器将其扩展为某个默认的主页。
21. 可以使用Linux的TELNET程序来和因特网上的任何Web服务器执行事务，一个HTTP请求有一个请求行、若干个请求报头再跟随一个空的文本行来终止报头列表。一个请求行的形式是method URI version，HTTP支持很多种method，包括GET,POST,OPTIONS,HEAD,PUT,DELETE和TRACE，大多数HTTP请求是GET方法，其指导服务器生成和返回URI（统一资源标识符）标识的内容，URI是相应URL的后缀，包括文件名和可选的参数（如果用代理服务器请求内容，那么URI必须是完整的URL），version表明该请求遵循的HTTP版本。
22. 请求报头为服务器提供额外信息，例如浏览器的商标名或浏览器理解的MIME类型，其格式为header-name:header-data，Host报头在HTTP/1.1请求中需要而在1.0请求中不需要，代理缓存会使用Host报头。
23. HTTP响应和HTTP请求类似，一个HTTP响应同样有一个响应行，跟随若干个响应报头，再跟随一个空行，在跟随一个响应主体，一个响应行的格式是version status-code status-message，状态码是一个3位的正整数表示对请求的处理，状态消息给出与错误代码等价的英文描述，响应报头提供关于响应的附加信息，响应主题包含请求的内容
24. GET请求动态内容的参数中不允许出现空格，而必须用%20代替，服务器接受请求后fork一个子进程并在其上下文中execve执行CGI程序（或CGI脚本），在execve之前子进程将CGI环境变量设置为传递的参数，CGI可以通过在运行时设置环境变量来将其他信息传递给子进程，一个CGI将其动态内容发送到标准输出，在子进程execve之前其就用dup2函数将标准输出重定向到客户端相关联的已连接描述符，任何CGI程序写到标准输出的东西都会直接到达客户端，同时子进程要负责生成响应报头中的部分内容和终止报头的空行。
25. 标准I/O可以在CGI程序中正常工作，在子进程中运行的CGI程序不需要显式关闭输入输出流，子进程终止时内核会自动关闭所有描述符。
26. 使用select函数要求内核挂起进程，只有在一个或多个I/O事件发生后才将控制返回给应用程序，select函数处理类型为fd\_set的集合，相当于一组掩码，select函数会一直阻塞直到有一个描述符准备好可以读。select函数有副作用，会修改参数fdset指向的集合指明读集合的一个子集，称为准备好集合（ready set），这个集合由读集合中准备好可以读的描述符组成，因此在每次调用select时必须更新读集合。
27. select函数的一个问题是一旦其连接到某个客户端就会连续回送输入航直到客户端关闭这个连接中它那一端，对此可以用更细粒度的select，比如每次只传送一行。
28. 现代高性能服务器使用的都是基于I/O多路复用的编程方式，主要是基于性能优势，其优点是比基于进程的设计给了程序员更多对程序行为的控制，同时每个逻辑流都能访问该进程的全部地址空间，这让在流之间共享数据变得容易了，同时事件驱动设计比基于进程的设计更高效。但缺点是编码复杂并且不能充分利用多核处理器。
29. 线程由内核自动调度，内核通过每个线程唯一的整数线程ID识别线程，多个线程共享这个进程虚拟地址空间的所有内容：代码、数据、堆、共享库和打开的文件
30. 每个线程例程都以一个通用指针作为输入，并返回一个通用指针
31. 一个线程以下列方式之一终止：顶层线程例程终止、调用pthread\_exit（主线程会等待所有对等线程终止再终止主线程和整个进程）、某个对等线程调用exit终止进程、另一个对等线程以当前线程ID作为参数调用pthread\_cancel来终止当前线程
32. 线程通过调用pthread\_join等待其他线程终止，只能等待一个指定的线程终止
33. 一个可结合的线程能被其他线程回收和杀死，在回收之前其内存资源不释放，而一个分离的线程是不能被其他线程回收或杀死的，其内存资源在终止时自动释放，线程默认被创建成可结合的，每个可结合线程都应被其他线程显示地收回，要么调用pthread\_detach函数被分离
34. 可以调用pthread\_once函数来初始化与线程例程相关的状态，其第一个参数once\_control总是被初始化为PTHREAD\_ONCE\_INIT，第一次以参数once\_control调用函数时会调用第二个参数对应的初始化函数，而以后的调用就不会做什么。
35. 一个变量是共享的，当且仅当多个线程引用这个变量的某个实例，寄存器从不共享，而虚拟内存总是共享。
36. 运行时虚拟内存的读写区域只包含每个全局变量的一个实例，任何线程都可以引用；每个线程的栈都包含它所有本地自动变量（即定义在函数内部但没有static属性的变量）的实例；虚拟内存的读写区域是包含在程序中声明的每个本地静态变量的一个实例，每个对等线程都读写这个实例。本地自动变量也是可以被共享的。
37. 操作共享变量的指令(L,U,S)构成了一个临界区，这个临界区不应和其他进程的临界区交替执行，我们要确保每个进程在执行其临界区的指令时，拥有对共享变量的互斥的访问。两个临界区的交集形成的状态空间称为不安全区，不安全区与其交界的状态相毗邻，但并不包括这些状态，绕开不安全区的轨迹线叫安全轨迹线，接触到任何不安全区的轨迹线叫不安全轨迹线
38. V必须重启一个正在等待的线程，当有多个线程在等待同一个信号量时，不能预测V操作要重启哪一个线程。P和V操作保证了信号量不会有负值，这称为信号量不变性。
39. sem\_init将信号量初始化，sem\_wait和sem\_post用来执行P，V操作，通常将其封装成P,V函数
40. 用P(s)和V(s)操作将相应临界区包围来保护共享变量的信号量叫二元信号量，其值总是0,1，以提供互斥为目的的二元信号量也称为互斥锁，P一个互斥锁称为互斥锁加锁，反之为互斥锁解锁。一个可以用作一组可用资源的计数器的信号量称为计数信号量。
41. 进度图不能处理在多处理器上的并发执行。
42. 第一类读者-写者问题读者优先，第二类写者优先。
43. 基于预线程化的并发服务器由一个主线程和一组工作线程组成，主线程不断接受来自客户端的连接请求，将得到的链接描述符放在一个有限缓冲区中，每个工作线程凡服从共享缓冲区中取出描述符，为客户端服务，然后等待下一个描述符。
44. 并发线程通常在多核处理器上运行的更快，因为操作系统在多个核上并行地调度这些并发线程，但由于同步操作（P、V）开销太大，因此可能会有越多线程并行，性能越差，因此必须用尽可能多的有用计算来弥补同步的开销。同时，随线程数量的增加，运行时间实际上会增加一点，这是由于在一个核上多个线程上下文切换的开销，因此并行程序常被写为每个核上只运行一个线程。
45. 一个函数是线程安全的，当且仅当被多个并发线程反复调用时会一直产生正确的结果。线程不安全的函数主要包括：不保护共享变量的函数（可以通过同步操作保护共享变量来编写线程安全的版本，但增大了同步开销）；保持跨越多个调用的状态的函数（如伪随机数生成器rand，使其线程安全的唯一方法是重写使之不使用任何static变量而是依靠调用者在参数中传递状态消息）；返回指向静态变量的指针的函数（被一个线程使用的结果可能被另一个线程修改了，因此可以重写函数或使用加锁-复制技术，定义一个包装函数对互斥锁加锁，然后把返回的结果复制到一个私有的内存位置，然后对互斥锁解锁）；调用线程不安全函数的函数（不一定，如果是1类或3类，那么可以用互斥锁保护调用位置与共享数据即可，但如果是第二类那么也是线程不安全的）。
46. 可重入函数指在被多个线程调用时不会引用任何共享数据，但不是所有的线程安全函数都是可重入的，同时可重入函数通常比不可重入的线程安全函数高效，因为它们不需要同步操，将第二类线程不安全函数转化为线程安全函数的唯一方法就是重写它们使之称为可重入的。如果所有的函数参数都是值传递的，并且所有的数据引用都是本地自动栈变量，那么函数就是显式可重入的；但如果允许使用指针传递参数并且小心传递非共享变量的指针，那么其是隐式可重入的，可重入性有时既是被调用者也是调用者的属性，并不是被调用者单独的属性。Linux系统提供大多数线程不安全函数的可重入版本，可重入版本的名字总是以\_r后缀结尾。
47. 当一个程序的正确性依赖于一个线程要在另一个线程到达y点之前到达自己的x点时就会发生竞争。死锁在进度图上会显示为两个线程的禁止区重叠，互斥锁加锁的顺序规则是给定所有的互斥操作的一个全序，如果每个线程都是以一种顺序获得互斥锁并以相反的顺序释放，那么这个程序就是无死锁的。