ICS 第八章作业

3.

（1）程序的功能是在标准输出设备（文件描述符为 stdout）上显示字符串“Hello，world.”。

（2）执行到第 16、20 行的“int $ 0x80”指令时从用户态转到内核态执行。

（3）该用户程序第 16 行指令调用了 4 号系统调用 write，对应服务例程为 sys__write（）函数；第 20 行指令调用了 1 号系统调用 exit，对应服务例程为 sys__exit（）函数。

5.

（1）因为 hcllo．c 中使用了 C 标准库函数 printfO，所以需在 hello．c 文件的开头加“＃include < stdio.h > ”。因为在 stdio.h 头文件中有 printfo）函数的原型声明，并且 printfO）函数是 C 标准库函数，所以，虽然 hello.c 中没有定义 printf（）函数，也没有它的原型声明，但是通过对 hello.c 和 stdio.h 的预处理，编译器会得到 printf（）的原型声明，从而获得符号 printf 的相关信息，使得链接器进行链接的时候，能够从标准 C 库（libc.a 或 libc.so）中得到 printf 模块的信息，完成链接工作

（2）需要经过预处理、编译、汇编、链接才能生成可执行文件 hello，然后通过启动 hello 程序执行。预处理阶段主要是对带＃的语句进行处理；编译阶段主要是将预处理后的源程序文件编译生成汇编语言程序；汇编阶段主要是将汇编语言源程序转换为可重定位的机器语言目标代码文件；链接阶段将多个可重定位的机器语言目标代码以及库函数链接起来，生成最终的可执行文件。

（3）因为 printf（）函数默认的输出设备为标准输出设备 stdout，所以无需指定字符串的输出目的地。执行 hello 程序后，自动会在 stdout 设备（屏幕上）显示字符串。

（4）字符串 " Hello，world．\ n " 在机器中对应的 0 / 1 序列（机器码）是该字符串中每个字符对应的 ASCII 码，即“48H 65H 6CH 6CH 6FH 2CH 77H 6FH 72H 6CH 64H 0AH 00H”。这个 0 / 1 序列存放在 hello．o 文件的 .rodata 节中，作为只读数据使用，所以从第 2 题图 8.1 中的汇编指令“movl ＄0x8048510,0x4（％esp）”可以看到，字符串的首地址是一个绝对地址 290

0x8048510，这个 0 / 1 序列在可执行目标文件 hello 的只读代码段（read-only code segment）中。

（5）若采用静态链接，则需要用到 printf．o 模块来解析 hello．o 中的外部引用符号 printf，printf.o 模块在静态库 libc.a 中。静态链接后，printf.o 中的代码部分（.text 节）被映射到虚拟地址空间的只读代码段（read-only code segment）中。若采用动态链接，则函数 printf（）的代码在虚拟地址空间中的共享库映射区。

（6）write（）函数的调用语句为：

write(1, "Hello, world.\n", 14);

字符串首地址

参数按照从右向左的顺序压栈，即 14 先压栈，然后是字符串的首地址压栈，最后是 fd = 1 压栈。随后执行 call 指令，将返回地址压栈后跳转到 write 代码执行。在 write 代码中，第一条指令将 EBX 压栈。此时栈中的状态如图 8.7 所示。可以看出，在地址 R［esp］＋8 处存放了 fd = 1，在 R［esp］＋12 处存放了指向字符串 " Hello，world．\ n " 的指针，在 R［esp］＋16 处存放了 14。

从上述代码可以看出，该 Linux 系统中系统调用返回的最大错误号是 4095。因为当返回值大于等于 0xf001 时进行出错处理，显然，这些值在 0xfffff001（-4095）和 0xff（-1）之间，

通常错误号是正整数，因此需要对其取负，故错误号范围为 1 ~ 4095。
（7）显然，第 1 题和第 2 题给出的实现方式，其程序设计的便捷性和可移植性都不如本题给出的实现方式，第 1 题采用汇编程序设计方式，只要参数不同，就需要重新编写不同的指令；第 2 题采用 write 函数直接进行系统调用，只能在支持 write 系统调用的系统（即类 UNIX）中执行。而本题给出的是 C 库函数调用，所以可以在不同的系统中执行，只要该系统具有 C 语言程序设计环境即可，因而本题给出的程序的可移植性更好。
不过，第 1 题实现方式下的程序执行性能最好，因为用汇编实现时，省去了高级语言程序中大量的函数调用，因而它的执行时间最短。

8.
要达到打印机最快的打印速度，打印机的数据传输率应达到每分钟为 6x50 行 x80 字符 = 24000 字符，即打印机的数据传输率应为 24000 字符 / 60s = 400 字符 / s。
若采用中断方式打印字符，则最长应该每隔 1 / 400s = 2.5ms 处理一次中断申请。而实际的中断响应及处理时间仅为 1000x1 / 500MHzx1000 = 0.002ms。因为 0.002ms < < 2.5ms，所以可以采用中断方式进行字符打印输出。

10.
（1）由题意可知，中断源 1 的处理优先级最高，说明 1 号中断源对其他所有中断源都屏蔽，其屏蔽字为全 1；3 号中断源的处理优先级最低，所以除了 3 号中断源之外，对其他中断源全都开放，其屏蔽字为 00100。以此类推，得到所有各个中断源的中断服务程序中设置的中断屏蔽字
（2）在运行用户程序时，同时出现中断源 2 和 4，因为用户程序对所有中断源都开放，所以在中断响应优先级排队电路中，中断源 2 和 4 进行排队判优，根据中断响应优先级 2 > 4，因此先响应 2 号中断源。在 CPU 执行中断源 2 的中断服务程序过程中，首先保护现场、保护旧屏蔽字、设置新的屏蔽字 01100，然后在具体中断处理前先开中断。一旦开中断，则马上响应 4 号中断源，因为第 2 号的中断屏蔽字中对第 4 号中断源的屏蔽位是 0，也即对第 4 号中断源是开放的。在第 4 号中断处理结束后，回到第 2 号中断源的中断服务程序执行；在具体处理第 2 号中断过程中，同时发生了第 1、3、5 号中断源请求，因为第 2 号中断对第 1、5 号中断开放，对第 3 号中断屏蔽，所以只有第 1 和第 5 两个中断源进行排队判优，根据中断响应优先级 1 > 5，所以先响应第 1 号中断源。因为第 1 号中断源的中断处理优先级最高，所以在其处理过程中不会响应任何新的中断请求，直到中断处理结束，然后返回第 2 号中断源；因为第 2 号中断源对第 5 号中断开放，所以在第 2 号中断源的中断服务程序中执行一条指令 295 后，又转去执行第 5 号中断源的中断服务程序，执行完后回到第 2 号中断源，在第 2 号中断源的中断服务程序执行过程中，虽然第 3 号中断源有中断请求，但是，因为第 2 号中断源对第 3 号中断源不开放，所以第 3 号中断源一直得不到响应。直到第 2 号中断源处理完回到用户程序，才能响应并处理第 3 号中断源的请求。