1. 以下选项给出的程序中，不在内核态运行的是（ A ）。

A. 命令行解释程序　　B. 设备驱动程序　　C. 系统调用服务例程　　D. 中断服务程序

2. 以下选项给出了几个在 Linux 系统的程序中使用的函数，其中在内核态运行的是（ B ）。

A. fread()　　B. sys_read()　　C. read()　　D. _fillbuf()

3. 假定有一个事务处理系统 A，其处理器的速度为每秒钟执行 5 千万条指令，每个事务需要 5 次 I/O 操作，每次 I/O 操作需要 10 000 条指令。如果系统 A 每秒钟最多发出 1000 次 I/O 操作。则它每秒钟处理的事务数最多能达到（ A ）个。（忽略延迟并假定事务可以不受限制地并行处理）

A. 200　　B. 1000　　C. 2000　　D. 10000

4. 在采用中断方式进行打印控制时，在打印控制接口和打印部件之间交换的信息不包括以下的 （ D ）。

A. 打印字符点阵信息　B. 打印控制信息　　C. 打印机状态信息　　D. 中断请求信号

5. 主机和外设之间的正确连接通路是（ A ）。

A. CPU 和主存 －I/O 总线 －I/O 接口(外设控制器)－ 通信总线(电缆)－ 外设

B. CPU 和主存 －I/O 接口(外设控制器)－I/O 总线 － 通信总线(电缆)－ 外设

C. CPU 和主存 －I/O 总线 － 通信总线(电缆)－I/O 接口(外设控制器)－ 外设

D. CPU 和主存 －I/O 接口(外设控制器)－ 通信总线(电缆)－I/O 总线 － 外设

6. 以下是有关程序直接控制（查询）I/O 方式的叙述：

① 无条件传送接口中不记录状态，无需状态查询，可直接定时访问

② 条件传送接口中有“就绪”、“完成”等状态，可定时查询或独占查询

③ 通过 CPU 执行相应的无条件传送程序或查询程序来完成数据传送

④ 适合于巡回检测采样系统或过程控制系统，以及非随机启动的字符型设备

以上叙述中，正确的有（ D ）。

A. 仅①和②　　B. 仅①和②和③　　C. 仅①和③和④　　D. 全部

7. 以下关于 I/O 子系统的描述中，错误的是（ D ）。

A. I/O 子系统包含 I/O 软件和 I/O 硬件两大部分

B. I/O 软件包含用户空间 I/O 软件部分和内核空间 I/O 软件部分

C. 内核空间 I/O 软件包含设备无关软件、设备驱动程序和中断服务程序

D. 直接控制 I/O 硬件的是设备驱动程序，而不是中断服务程序

8. 以下是有关对 DMA 方式的叙述：

① DMA 控制器向 CPU 请求的是总线使用权

② DMA 方式可用于键盘和鼠标器的数据输入

③ DMA 方式下整个 I/O 过程完全不需要 CPU 介入

④ DMA 方式需要用中断处理进行辅助操作

以上叙述中，错误的是（ B ）。

A. 仅①和② B. 仅②和③ C. 仅②和④ D. 仅③和④

简答题：

1. 有以下 C 语言代码：

```
1    #include <stdio.h>
2    int main()
3    {
4        printf( "Hello, world.\n");
5    }
```

假定源程序文件名为 hello.c，可重定位目标文件名为 hello.o，可执行目标文件名为 hello，程序用 GCC 编译驱动程序处理，在 IA-32/Linux 系统中执行。回答下列问题或完成下列任务。

（1）为什么在 hello.c 的开头需加“#include <stdio.h>”？ 为什么 hello.c 中没有定义 printf()函数，也没它的原型声明，但 main()函数引用它时没有发生错误？

（2）需要经过哪些步骤才能在机器上执行 hello 程序？说明各个环节的处理过程。

（3）为什么 printf()函数中没有指定字符串的输出目的地，但执行 hello 程序后会在屏幕上显示字符串？

（4）字符串"Hello, world.\n"在可执行目标文件 hello 的哪个段中？

（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 模块在哪个静态库中？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哪个段中？若采用动态链接，则函数 printf()的代码在虚拟地址空间中的何处？

答：

（1）因为 hello.c 中使用了 C 标准库函数 printf()，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 printf()函数的原型声明，并且 printf()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 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。