武汉大学国际软件学院 2017-2018 学年第一学期期末考试试卷

A 卷

课程名称：《 系统级程序设计 》
年级：_____专业：_____ 专业方向：_____ 层次：本科
姓名：_____学号：_____考分：_____

说明：1、答案一律书写在答题纸上，书写在试卷上或其他地方一律无效。
2、请准确规范书写姓名和学号，否则作废。

一、 填空题（共 20 分）

1. 请用括号“( )”括出语句 “c = getchar() != EOF”的优先级顺序：_____。（1 分）
c = （getchar() ）!= EOF _____。

2. 在标准 C 语言中，符号通常有多重含义，依赖于对上下文的理解。比如，符号“*”在用于声明时，可以理解为_____，而用于数学表达时，则表示乘法操作符；而“&”有时用于表达“取地址操作符”，而用于位操作时，则表示_____。（各 1 分）
1、指针； 2、与操作

3. 请写出 32 位系统中，标准 C 所能表达的最大正整数的 16 进制形式_____。（1 分）
7fffffff

4. 用英语填充以下对声明“char* const *(*next) () ”解释的缺失部分：next is a _____ to a function returning a _____ to a const pointer-to-char （各 1 分）【提醒：用中文回答得一半分】
1、pointer; 2、pointer

5. 在现代计算机的内存布局体系中，运行时堆栈通常位于内存的_____地址位置，而运行时堆则相对地位于内存的_____地址位置。（各 1 分）
1、高； 2、低

6. 假设在 Linux 操作系统中，使用 gcc 作为编译器。对于标准 C 程序 hello.c 源码，经过预处理过程后，将转变为 modified source program 形式 hello.i，经过编译器过程后，将转化为_____形式 hello.s，经过汇编过程后，将转变为_____形式 hello.o，经过链接过程后，将转变为最终的 executable object program 形式 hello。（各 1 分）【提醒：使用英语回答，用中文回答得一半分】
1、assembly program; 2、relocatable object program

7. 在线程存储模型中，每个线程拥有独立的线程上下文，其组成包括：线程 ID、堆栈、堆栈指针、_____、条件代码、_____（各 1 分）

1、程序计数器；　2、通用寄存器值

8. 在对堆内存的操作函数中，malloc()与 calloc()的主要差异是_____，而 sbrk(0)返回的是_____（各 2 分）

1、malloc()只对堆进行分配操作，而 calloc()除了堆堆进行分配外，还进行了初始化操作；　2、当前堆的 brk 指针的地址

9. 在 32 位系统下，对于某个位于 k+1 层的 Cache 结构，假设其为直接映射型（Direct Mapped Cache），如果已知其容量为 1024bytes，Cache Line 块大小为 16 bytes，那么，此层的 Cache 有_____个 sets，其对应的上层 k 的地址构成中，t、s、b 分别为_____位长、_____长位，以及_____位长。（各 1 分）

1、64 个；
2、22 位
3、6 位
4、4 位

二、 简答题（每题 5 分，共 25 分）

1、 解释未初始化的局部变量与已初始化局部变量在内存表现上有何相同点与不同点？

相同点：未初始化的局部变量和已初始化的局部变量均存储在堆栈的区域。（2.5 分）

不同点：未初始化的局部变量采用统一的标识（Windows 系统中），或随机化的数值（非 Windows 系统）；已初始化的具备变量即为自身的初始化值。（2.5 分）

2、 什么是空间局部性？以一个典型的代码例子说明空间局部性在 C 语言中是普遍存在的。

应用程序倾向于引用的数据项临近于其他最近引用过的数据项，或者临近于最近自我引用过的数据项，称为局部性。（可以不写）

应用程序在不远的将来反复引用某存储的临近位置，称为空间局部性。（2 分）

典型的空间局部性的例子最好是对数组的顺序引用，如 for 循环中，以步长为 1 或接近于 1 的小值反复对数组引用：（3 分）

```
int sum=0;
for (int i=0; i<100; i++)
{
 sum += a[i];
```

}

扣分原则：划线的未表达各扣 1 分

3、 对于 C 函数原型：void *malloc (size_t size)，翻译下面的一段话，并就其中划线的部分解释为什么要这样定义。

The *malloc* function returns a pointer to a block of memory of at least size bytes[1] that is suitably aligned[2] for any kind of data object[3] that might be contained in the block.

翻译：*malloc 函数返回一个内存块的指针，该内存块可以用于以块形式链接的任意类型数据，且块以合适的方式进行对齐，同时，块的大小等于或大于所申请的大小（以 byte 为单位）。（2 分）*

1: 由于对其的原因，所申请的分配大小是最小容量，实际分配的结果可能大于所申请的大小

2: 针对运行系统、编译器的不同，块对齐的策略可能不同，因此定义中仅规定了对齐的原则（即 suitable），未给出对齐的具体策略。

*3: 由于 malloc 函数的返回类型是 void *，所以，其存储的数据内容可以是任意类型的。*

（以上各 1 分）

4、 **在自行构建一个堆的分配器（Allocator）时，需要考虑的策略有哪些？**

自由块（free block）（或未分配块）的组织策略：怎样保证对自由块的追踪；

安置策略：对即将分配的块，如何选择合适的自由块进行安置；

拆分（或分裂）策略：对于新分配的块小于所容纳的自由块大小的情形，如何处理所剩余的自由块；

连接（或合并）策略：对于新释放的自由块，如何根据其前后的块的状态进行合并，以形成较大的自由块。

扣分原则：冒号前未写的，扣 0.5 分；划线部分未表达的，各扣 0.5 分

5、 现有如下代码，（1）、请写出其运行结果；（2）、这种现象在 C 语言中被称为什么？

```
Switch (2) {
    case 1: printf (“case 1 \n”);
    case 2: printf (“case 2 \n”);
    case 3: printf (“case 3 \n”);
    case 4: printf (“case 4 \n”);
    default: printf (“default \n”);
}
```

（1） 一运行结果为：

```
case 2
case 3
case 4
default
```

（答错扣 3 分）

（2） *一这种现象在 C 语言中称之为“Fall Through”现象。中文可翻译为“跌落”。*（答错扣 2 分。改卷注意：写成中文，只要意思对也给分）

三、 实践题（每题 10 分，共 30 分）

1、 写出 Amdahl’s Law 的形式化表达式，注明各个符号的含义，并阐明其对代码性能优化作业的指导意义。（10 分）

$$T_{New} = (1-\alpha)T_{Old} + (\alpha T_{Old})/k$$

或

$$S = \frac{T_{Old}}{T_{New}} = \frac{1}{(1-\alpha)+\alpha/k}$$

（2 分）

其中：

T_{new} 代表优化后的应用程序运行时间

T_{old} 代表应用程序优化前的运行时间

a 代表应用程序优化的比例

k 代表优化部分性能提高的倍数

S 代表应用程序优化前后性能整体提升的比例（此句可以不写）

（共 4 分，缺 1 个扣 1 分）

指导意义：

1、所有的应用程序都存在优化的空间（可能性）

2、对应用程序的少量的优化可以换来性能的大幅度提升

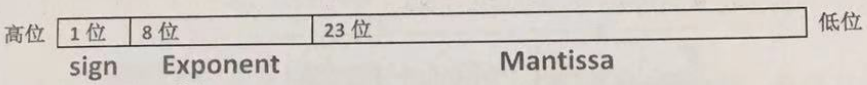
3、如果应用程序仅仅只有少量的部分可以优化，那么，未优化部分的比例决定了应用程序整体的性能提升的幅度

4、知道如何优化是一个重要的技能；但是，知道何时应该停止优化同样是非常重要的

（以上各 1 分）

2、 **1）请绘图表示 IEEE 规定的单精度浮点数的三部分结构，标明各自部分的名称（英文）以及对应的位长。2）结合图解释为什么在 C 语言中，当强行将 double 类型转换为 float 类型时，有被理解为正负无穷大的可能。（10 分）**

1）绘图：



记分原则（共 6 分）：

- 名称未写：扣 3 分；位长未标：扣 3 分；
- 名称未写对，或用中文，扣 1 分；位长未标对，扣 1 分；
- 未标注高地位，不扣分；

2）解释

（1） 结合上图，“±∞”在 float 类型中定义为：指数部分（E，占据 8 位）全为 1，同时底数部分（M，占据 23 位）全为 0.（不给出±∞的说明，扣 2 分；给出了，但是答错扣 1 分）

（2） 如果将 double 型强制转换为 float 型后，由于 double 类型的 E、M 位数均大于 float 类型，在特殊情况下由于溢出的原因，可能会导致指数部分（E）全为 1，同时底数部分（M）全为 0，也就是被理解为“±∞”。（答错扣 2 分）

3、 阅读如下代码，假设在 32 位系统情况下，用内存布局图绘图表示函数 find( ) 调用时，堆栈中参数、局部变量、栈顶指针、栈底指针、返回地址的相对分配序列。（提醒：题目要求是在 32 位系统假设下）

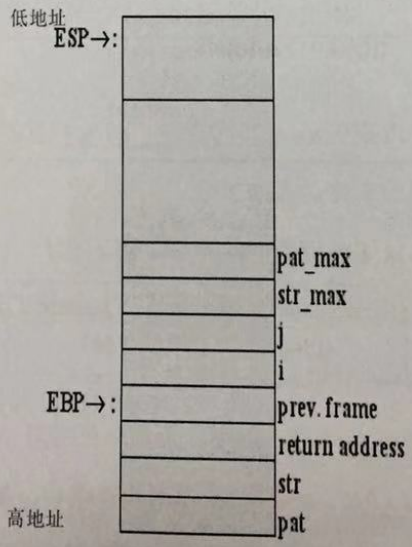
```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

int find(char *str, char *pat)
{
    int i, j, str_max, pat_max;
    pat_max = (int)strlen(pat);
    str_max = (int)strlen(str) - pat_max;
    for (i = 0; i < str_max; i++) {
        for (j = 0; j < pat_max; j++) {
            if (str[i + j] != pat[j]) break;
        }
        if (j == pat_max) return i;
    }
    return -1;
}

void main()
{
    printf("find(\"this is a test\", \"is\") -> %d\n",
           find("this is a test", "is"));

    printf("find(\"this is a test\", \"IS\" -> %d\n",
           find("this is a test", "IS"));
    getchar();
}
```

分配序列如下：



扣分原则：

1)、以上词条，缺 1 个扣 1 分；其中，prev. frame 也可用其它的名字（或者中文），比如：saved ebp，调用者 ebp 等；

2)、未标明高低地址方向的，扣 1 分；

3)、顺序错误的，视情况扣 3~8 分

四、 综合题（共 25 分）

1、阅读如下代码，回答：

（1）、该代码的功能是什么？（2 分）

（2）、该代码使用的逻辑操作符（^）的英文名称是什么？（2 分）

（3）、若将参数类型转换为 float，该代码还正确吗？理由是什么？（2 分）

（4）、若要保障该代码的安全调用，需设定哪些限制条件？（至少正确回答 2 个）（4 分）

```
void swap ( int *x, int *y)
{
        *x = *x ^ *y;
        *y = *x ^ *y;
        *x = *x ^ *y;
}
```

（1）、该代码对内存中 x、y 指向地址中的内容进行了互换

扣分原则：划线的未表达各扣 2 分

（2）、exclusive OR【如用中文答对了，给 1 分】 （2 分）

（3）、不正确。因为位操作只能在整数之间进行。（2 分）

（4）、1）、x y 不能为 NULL

2)、x y 不能指向同一地址，否则返回的两个结果均为 0

3）、x y 中任何一个不能存储在寄存器中，因为无法取得寄存器的地址

4)、x y 其中如何一个不能是数组，否则同样错误

（以上 4 点，只要回答任意两个即可，每个 2 分）

2、现有如下代码片段。为简单起见，假设采用直接映射型缓存结构，缓存块的大小为 8bytes（即可以存放 2 个整数），而且缓存有两个 set 构成，即共有 16bytes 的缓存空间。假设变量 i、j 存储在 register 中：

（1） 填写表格中空出部分的相关值，其中命中以符号 h 表示，不命中以符号 m 表示；（8 分）

（2） 在尽量少变动源码的前提下，改写该段代码，提高该段代码的内存命中率（3 分）

（3） 参照该表格，另画一张表表示改写后的代码的命中情况，并计算改写前、后的该段代码的内存命中率。（4 分）

```
typedef int array[2][2];
void transposel(array dst, array src)
{
   int i, j;
   for (i=0; i<2; i++)
        for (j=0; j<2; j++){
            dst[j][i]= src[i][j];
        }
   return;
}
```

1、 改写前：

src[0][0]	src[0][1]	src[1][0]	src[1][1]	dst[0][0]	dst[0][1]	dst[1][0]	dst[1][1]
Set 0		Set 1		Set 0		Set 1	

【上面的表格可以不画】

变量索引		命中状态		当前缓存中的变量	
i	j	src[][]	dst[][]	Set 0	Set 1

0	0	Cold m	m	dst[0][0]; dst[0][1]	--
0	1	m	Cold m	src[0][0]; src[0][1]	dst[1][0]; dst[1][1]
1	0	m	m	dst[0][0]; dst[0][1]	src[1][0]; src[1][1]
1	1	h	m	dst[0][0]; dst[0][1]	dst[1][0]; dst[1][1]

打分原则：

1、每个空格 0.5 分。右半部分的写法可以不同，如写为 dst00;dst01，

或者：dst_{00}; dst_{01} 均可；

2、没有写 cold，不扣分；

2 和 3、有多种改写的可能性：

	dst[j][i]=src[i][j]	dst[1-j][i]=src[i][1-j]
src[2][2]	A【原始代码】	B
src[3][2]	C	D

情况 B:

src[0][0]	src[0][1]	src[1][0]	src[1][1]	dst[0][0]	dst[0][1]	dst[1][0]	dst[1][1]
Set 0		Set 1		Set 0		Set 1	

变量索引		命中状态		当前缓存中的变量	
i	j	src[][]	dst[][]	Set 0	Set 1
0	0	Cold m	Cold m	src[0][0]; src[0][1]	dst[1][0]; dst[1][1]
0	1	h	m	dst[0][0]; dst[0][1]	dst[1][0]; dst[1][1]
1	0	m	m	dst[0][0]; dst[0][1]	dst[1][0]; dst[1][1]
1	1	m	h	dst[0][0]; dst[0][1]	src[1][0]; src[1][1]

情况 C: (scr[3][2])

src[0][0]	src[0][1]	src[1][0]	src[1][1]	src[2][0]	src[2][1]	dst[0][0]	dst[0][1]	dst[1][0]	dst[1][1]
Set 0		Set 1		Set 0		Set 1		Set 0	

变量索引		命中状态		当前缓存中的变量	
i	j	src[][]	dst[][]	Set 0	Set 1
0	0	Cold m	Cold m	src[0][0]; src[0][1]	dst[0][0]; dst[0][1]
0	1	h	m	dst[1][0]; dst[1][1]	dst[0][0]; dst[0][1]
1	0	m	m	dst[1][0]; dst[1][1]	dst[0][0]; dst[0][1]
1	1	m	h	dst[1][0]; dst[1][1]	src[1][0]; src[1][1]

情况 D:

src[0][0]	src[0][1]	src[1][0]	src[1][1]	src[2][0]	src[2][1]	dst[0][0]	dst[0][1]	dst[1][0]	dst[1][1]
Set 0		Set 1		Set 0		Set 1		Set 0	

变量索引		命中状态		当前缓存中的变量	
i	j	src[][]	dst[][]	Set 0	Set 1
0	0	Cold m	m	dst[1][0]; dst[1][1]	--
0	1	m	Cold m	dst[1][0]; dst[1][1]	dst[0][0]; dst[0][1]
1	0	m	h	dst[1][0]; dst[1][1]	src[1][0]; src[1][1]
1	1	h	m	dst[1][0]; dst[1][1]	dst[0][0]; dst[0][1]

B、C、D 三种情况下，改写后的命中率都是 2/8=25%

改写前的命中率：1/8=12.5%

扣分原则：

1、B、C、D 只要答对一种即可。如有其它的解决方案，给我来判断；

2、对于（3），如果没有画表，扣 1 分；

3、对于（3），画对表了，但是命中率没有算对，扣 2 分；算对了，也画表了，但是没有画对，扣 0.5 分。