

清华大学本科生考试试题专用纸

考试课程：操作系统（B 卷）

时间：2015 年 04 月 13 日下午 1:30~3:05

系别：_____ 班级：_____ 学号：_____ 姓名：_____

- 答卷注意事项：
1. 答题前，请先在试题纸和答卷本上写明 A 卷或 B 卷、系别、班级、学号和姓名。
 2. 在答卷本上答题时，要写明题号，不必抄题。
 3. 答题时，要书写清楚和整洁。
 4. 请注意回答所有试题。本试卷有 6 个题目，共 13 页。
 5. 考试完毕，必须将试题纸和答卷本一起交回。

一、（10 分）如果 80386 机器的一条机器指令(指令长 8 个字节)，其功能是把一个 32 位字的数据装入寄存器，指令本身包含了要装入的字所在的 32 位地址。指令和数据的地址可不按 8 字节或 4 字节对齐。这个过程最多会引起几次缺页中断？请说明理由。

二、（20 分）uCore OS 在启动过程中，经历了多次地址映射的变化，理解这些变化对使用 gdb 调试 uCore 启动初期的代码十分重要。下面的描述中假定你已完成实验二的练习。

- 1) (启动) 对于 x86 机器，加电后处于实模式下，并且 cs 寄存器的段选择子部分为 0xf000, 隐藏的基址部分为 0xffff0000, eip 寄存器为 0xffff0, 则此时待执行的指令的物理地址为(---1---)。
- 2) (16 位实模式) 执行完第一条 ljmp 指令后, cs 寄存器的段选择子部分变为 0xf000, eip 变为 0xe05b, 则此时待执行指令的物理地址为 (---2---)。
- 3) (进入 32 位保护模式) BIOS 会将 ucore bootloader 加载到物理地址 0x7c00, 且设置 cs 为 0x0, eip 为 0x7c00. 在 bootloader 中, 使用 lgdt 加载了如下全局描述符表, 然后进入 32 位保护模式. 此时为了正常工作, 应该用 ljmp 将 cs 设置为 (---3.1---), 将 ds/es/fs/gs/ss 设置为 (---3.2---). 此时若 eip 为 0x7c6d, 则实际被执行指令的物理地址为 (---3.3---)。

注：一个段描述符占 8Byte.

```
SEG_NULLASM                                # null seg
SEG_ASM(STA_X|STA_R, 0x0, 0xffffffff) # code seg for bootloader and kernel
SEG_ASM(STA_W, 0x0, 0xffffffff)      # data seg for bootloader and kernel
```

当 bootloader 将其加载到内存时，使用的逻辑地址是 (---3.4---)，对应的线性地址是 (---3.5---)，对应的物理地址是(---3.6---)。

- 4) (32 位保护模式，调整段映射) 根据 tools/kernel.ld, ucore kernel 的入口点 kern_entry 的逻辑地址为 0xC0100000。ucore bootloader 跳转到 ucore kernel 的 kern_entry 执行后, ucore kernel 首先加载了新的全局描述符表:

```
#define KERNBASE 0xC000 0000
SEG_NULL
SEG_ASM(STA_X | STA_R, - KERNBASE, 0xFFFFFFFF) # code segment
SEG_ASM(STA_W, - KERNBASE, 0xFFFFFFFF)        # data segment
```

完成加载后, 访问逻辑地址 0xC010002A, 对应的线性地址是(---4.1---), 对应物理地址 (---4.2---).

5) (32 位保护模式 建立分页机制) 在 ucore kernel 初始化过程中, boot_map_segment 函数 建立了初始的二级页表内容, 可将线性地址 0xC0000000 映射到物理地址 0x0. 但是考虑到使能分页机制之后, 段映射和页映射同时生效, 所以需要在使能分页机制前, 建立页映射关系, 使得可将线性地址空间 0~4MB 映射到物理地址空间 0~4MB, 且 ucore kernel 在使能了分页机制之后, 还需紧接着最后一次加载新的全局描述符表, 并取消将线性地址空间 0~4M 映射到物理地址空间 0~4M, 从而才算完成了分页机制的建立。

```
SEG_NULL,
```

```
[SEG_KTEXT] = SEG(STA_X | STA_R, 0x0, 0xFFFFFFFF, DPL_KERNEL),
```

```
[SEG_KDATA] = SEG(STA_W, 0x0, 0xFFFFFFFF, DPL_KERNEL),
```

```
[SEG_UTEXT] = SEG(STA_X | STA_R, 0x0, 0xFFFFFFFF, DPL_USER),
```

```
[SEG_UDATA] = SEG(STA_W, 0x0, 0xFFFFFFFF, DPL_USER),
```

```
[SEG_TSS] = SEG_NULL,
```

在 ucore kernel 建立页机制过程中, 使能分页机制前, 逻辑地址 0xC010 0000 对应线性地址 (---5.1---) 和物理地址 (---5.2---);

使能了分页机制之后, 完成分页机制建立之前, 逻辑地址 0xC010 0000 对应线性地址 (---5.3---) 和物理地址 (---5.4---);

完成分页机制建立之后, 逻辑地址 0xC010 0000 对应线性地址 (---5.5---) 和物理地址 (---5.6---);

三、(20 分) 1) 请描述时钟 (clock) 置换算法和先进先出置换算法的工作原理。

2) 假定一个进程的虚拟存储访问序列如下。

a, b, c, d, a, b, e, a, b, c, d, e

请分别计算上述进程在分配 3 个物理页帧和 4 个物理页帧时的缺页次数。要求给出计算过程。

3) 什么是 belady 现象? 先进先出置换算法存在 belady 现象吗? 时钟置换算法存在 belady 现象吗? 如果存在, 请尝试给出反例。如果不存在, 请给出解释。

四、(20 分) 下面是与进程切换相关的 ucore 代码。请基于代码分析回答下面问题。

1) switch_to 函数的返回地址是放在哪个进程的堆栈中的?

2) 在 switch.S 文件中执行时, 切换前和切换后的进程的上下文(context)首地址在哪?

3) 进程切换代码中, 完成页表切换的代码行是什么?

4) 进程切换代码中, 完成内核堆栈切换的代码行是什么?

代码如下:

```
// Saved registers for kernel context switches.
```

```
// Don't need to save all the %fs etc. segment registers,
```

```
// because they are constant across kernel contexts.
```

```
// Save all the regular registers so we don't need to care
```

```
// which are caller save, but not the return register %eax.
```

```
// (Not saving %eax just simplifies the switching code.)
```

```
// The layout of context must match code in switch.S.
```

```
struct context {
```

```

uint32_t eip;
uint32_t esp;
uint32_t ebx;
uint32_t ecx;
uint32_t edx;
uint32_t esi;
uint32_t edi;
uint32_t ebp;
};

#define PROC_NAME_LEN      15
#define MAX_PROCESS        4096
#define MAX_PID            (MAX_PROCESS * 2)

extern list_entry_t proc_list;

struct proc_struct {
    enum proc_state state;           // Process state
    int pid;                        // Process ID
    int runs;                       // the running times of Proces
    uintptr_t kstack;               // Process kernel stack
    volatile bool need_resched;     // bool value: need to be rescheduled to release CPU?
    struct proc_struct *parent;     // the parent process
    struct mm_struct *mm;           // Process's memory management field
    struct context context;         // Switch here to run process
    struct trapframe *tf;           // Trap frame for current interrupt
    uintptr_t cr3;                  // CR3 register: the base addr of Page Directroy Table(PDT)
    uint32_t flags;                 // Process flag
    char name[PROC_NAME_LEN + 1];   // Process name
    list_entry_t list_link;         // Process link list
    list_entry_t hash_link;        // Process hash list
    int exit_code;                  // exit code (be sent to parent proc)
    uint32_t wait_state;            // waiting state
    struct proc_struct *cptr, *yptr, *optr; // relations between processes
};

static inline void lcr3(uintptr_t cr3)
{
    asm volatile ("mov %0, %%cr3:::r" (cr3));
}

```

```

}

/* *
 * load_esp0 - change the ESP0 in default task state segment,
 * so that we can use different kernel stack when we trap frame
 * user to kernel.
 * */
void
load_esp0(uintptr_t esp0) {
    ts.ts_esp0 = esp0;
}

void
proc_run(struct proc_struct *proc) {
    if (proc != current) {
        bool intr_flag;

        struct proc_struct *prev = current, *next = proc;
        local_intr_save(intr_flag);
        {
            current = proc;

            load_esp0(next->kstack + KSTACKSIZE);
            lcr3(next->cr3);

            switch_to(&(prev->context), &(next->context));
        }
        local_intr_restore(intr_flag);
    }
}

.text
.globl switch_to
switch_to:                                # switch_to(from, to)

    # save from's registers
    movl 4(%esp), %eax                    # eax points to from
    popl 0(%eax)                          # save eip !popl
    movl %esp, 4(%eax)
    movl %ebx, 8(%eax)
    movl %ecx, 12(%eax)
    movl %edx, 16(%eax)
    movl %esi, 20(%eax)
    movl %edi, 24(%eax)

```

```

movl %ebp, 28(%eax)

# restore to's registers
movl 4(%esp), %eax      # not 8(%esp): popped return address already
                        # eax now points to to

movl 28(%eax), %ebp
movl 24(%eax), %edi
movl 20(%eax), %esi
movl 16(%eax), %edx
movl 12(%eax), %ecx
movl 8(%eax), %ebx
movl 4(%eax), %esp

pushl 0(%eax)           # push eip
ret

```

五、(20 分) 有一台假想的计算机 x86-lite, 页大小 (page size) 为 32 Bytes, 支持 8KB 的虚拟地址空间 (virtual address space), 有 4KB 的物理内存空间 (physical memory), 采用二级页表, 一个页目录项 (page directory entry, PDE) 大小为 1 Byte, 一个页表项 (page-table entries PTEs) 大小为 1 Byte, 1 个页目录表大小为 32 Bytes, 1 个页表大小为 32 Bytes。页目录基址寄存器 (page directory base register, PDBR) 保存了页目录表的物理地址 (按页对齐)。在 x86-lite 计算机中的 4KB 的物理内存空间和 4KB 的 disk 空间的值, PDBR 的值如题后所示。

PTE 格式 (8 bit) : VALID | PFN6 ... PFN0 或者 VALID | DSN6 ... DSN0

PDE 格式 (8 bit) : VALID | PT6 ... PT0

其中:

VALID 为第 7 位

VALID==1 表示, 表示映射存在;

VALID==0 表示, 表示内存映射不存在, 并有 3 种情况:

a. 如果是 PTE, 且项的内容不为 0x7F, 则表示对应的物理页帧号 swap out 在硬盘上, 扇区号为物理页帧号;

b. 如果是 PTE, 且项的内容为 0x7F, 则表示既没有在内存中, 也没有在硬盘上

c. 如果是 PDE, 则表示既没有在内存中, 也没有在硬盘上

PFN 6..0 : 页帧号

DSN 6..0 : 硬盘扇区号

PT 6..0 : 页表的物理基址 >> 5

PDBR : 页目录基址寄存器, 保存了页目录表的物理地址 (按页对齐)。

1)、 请接合上述假想计算机 x86-lite, 描述二级页表机制的基本运行机理。

2)、 对于一个给定的 vaddr 值 (即一个虚地址) 和 paddr 值 (即页目录基址寄存器的值), 请用 C 语言表达式表示:

(2.1) vaddr 对应的页目录项 (PDE) 的 index

(2.2) vaddr 其对应的页表项 (PT) 的 index

(2.3) vaddr 对应的 paddr(物理地址值)

(2.4) 请回答下列 vaddr 虚地址是否有合法对应的物理内存, 请给出对应的 PDE index, PDE contents, PTE index, PTE contents, paddr(物理内存地址), contents of paddr (物理内存地址内容), 或 saddr(磁盘扇区地址), contents of saddr (磁盘扇区内容) 。

vaddr 137d

vaddr 01bb

vaddr 297d

vaddr 1c99

格式

vaddr 0c9e:

--> pde index:0x3 pde contents:(valid 1, pfn 0x6a)

--> pte index:0x4 pte contents:(valid 1, pfn 0x43)

--> To Physical Address 0x87e --> Value: 06

PDBR content: 0xf40 [This means the base address of page directory table is held in here]

```
-----
page 00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 01: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 02: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 03: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 04: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 05: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 06: 0a 0e 1b 1e 1b 13 11 06 12 05 09 19 11 16 0e 08 0f 11 07 1a 1e 05 07 19 16 06 12 07 1a 12 0f 05
page 07: b6 16 7f cf 7f d3 22 4c 97 2b 7f 7f 6c 7f f0 32 13 51 7f 72 dc c1 7f 7f 7f 7b 7f c2 7f 7e 3a 7f
page 08: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 09: 08 0b 1c 02 17 05 11 0c 0e 17 0c 03 03 02 1a 13 1d 15 00 14 18 16 0f 0b 0e 18 08 10 0e 1d 18 1c
page 0a: 7f f2 b5 57 44 7f 7f e1 d8 7f af 1e 28 d2 7f 35 21 a0 25 8c 1a 8b 03 7f 7f dd 7f 4b 7f ad 0e 23
page 0b: 05 0d 0c 14 19 0a 1c 14 0a 05 1d 14 01 15 0b 0d 17 06 08 10 1c 02 19 16 05 14 13 16 10 09 0f 07
page 0c: 07 05 15 0b 0b 18 07 19 13 0c 19 0a 1b 1c 0f 15 1d 17 17 1a 1d 17 1e 09 13 14 0b 0c 05 1d 0a 0e
page 0d: 0a 05 04 14 02 1e 14 01 1b 07 0e 11 04 0f 01 06 1c 19 10 15 1b 04 0f 04 03 03 06 01 06 0b 13 1a
page 0e: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 0f: 0b 13 0e 07 18 1d 14 10 04 0e 18 09 0d 0d 09 13 09 1c 19 09 02 00 19 0d 03 01 18 06 1c 0b 13 0a
page 10: 00 0a 0b 1c 1b 17 0d 10 07 19 0c 08 13 04 09 1c 02 04 06 07 0d 07 0a 07 07 12 0a 0b 17 01 04 1a
```

page 11: 00
page 12: 08 03 01 11 1e 00 1b 08 04 00 0a 02 0f 0e 05 1b 12 17 07 16 15 18 05 1d 0e 02 1c 09 1b 06 02 0f
page 13: 17 09 04 03 06 15 03 01 07 01 16 17 10 0a 01 16 07 11 1e 07 15 0d 10 0c 15 10 01 19 07 0a 0e 12
page 14: 02 19 04 0f 16 15 06 18 01 11 1b 10 09 1e 16 18 1b 1c 11 19 0f 07 08 08 00 13 0c 1c 01 18 06 0a
page 15: 00
page 16: 00
page 17: 15 0b 01 1e 04 01 0a 13 17 03 0a 00 0d 17 16 1b 17 1a 15 0e 06 14 09 03 0d 1b 03 12 0c 0f 04 1d
page 18: 00
page 19: 00
page 1a: 00
page 1b: 0e 10 0f 1d 1d 01 0c 10 14 10 19 03 0e 19 05 05 13 04 10 03 05 0a 18 12 12 13 0c 1e 12 13 02 07
page 1c: 00
page 1d: 00
page 1e: 00
page 1f: 00
page 20: 16 0a 0d 0c 19 00 11 04 04 12 0b 02 12 1c 13 0f 18 1c 04 09 1d 1c 06 15 1b 12 15 10 07 06 01 14
page 21: 07 09 06 1c 13 01 02 05 1e 0a 18 06 06 1d 02 0f 10 0a 17 1a 07 09 0f 0e 13 18 0a 1d 0e 00 0d 10
page 22: 00
page 23: 00
page 24: 00
page 25: 07 0f 08 0d 15 1c 12 19 02 0b 1e 04 0c 02 02 1b 1e 14 03 09 07 14 15 0d 10 03 10 1d 17 02 10 16
page 26: 00
page 27: 7c 8f df 39 d7 69 7f 7f 7f 7f 7f 7f 3e 7f 2c 7f 7f 9b 3f 3b 7f 7f a5 00 1f 7f 7f 7f 7f 7f 0c 7f
page 28: 00
page 29: 00
page 2a: 08 0f 00 02 17 05 17 18 0c 14 18 1a 04 05 0b 0e 09 00 11 1d 0b 10 0b 0d 1a 09 14 0e 10 1c 02 19
page 2b: 0c 12 17 1a 0a 12 13 15 12 0e 19 00 18 05 05 14 0e 0a 02 01 08 02 15 06 0b 0d 15 19 13 06 04 11
page 2c: 00
page 2d: 08 05 17 0f 10 19 00 10 0a 07 19 07 0c 0d 04 09 00 0f 13 1e 1d 0a 1d 03 04 10 15 12 01 0c 0f 0b
page 2e: 00
page 2f: 1e 0d 14 1b 01 09 0c 07 0b 15 04 15 09 14 05 18 0c 19 1d 12 00 0b 1e 14 17 0e 1d 1c 12 12 03 00
page 30: 00
page 31: 00
page 32: 00
page 33: 03 07 0a 0c 11 17 08 11 0a 07 04 1d 1c 19 05 04 03 09 0d 00 0d 15 19 05 00 18 0b 09 04 08 08 07
page 34: 00
page 35: 00 11 04 01 01 04 02 13 0f 1e 1c 1e 07 0d 07 12 13 18 15 07 0d 10 00 01 0c 03 16 07 03 05 07 06
page 36: 0e 0b 1e 11 10 0d 0d 1d 18 14 05 12 03 0a 00 15 1e 13 11 07 0d 0e 0d 08 06 17 1d 19 13 00 09 1a
page 37: 00
page 38: 00
page 39: 00
page 3a: 00
page 3b: 09 13 06 1c 1d 16 0d 0f 12 01 0c 10 05 02 18 0b 10 1c 12 08 1e 0b 00 15 03 09 1a 14 00 0d 0c 0f
page 3c: 0b 05 06 15 1c 1a 13 18 04 06 15 00 02 18 06 05 0c 19 00 1b 09 11 0e 03 1d 05 18 1b 0e 1d 01 04
page 3d: 04 05 0b 14 0d 06 0a 09 04 12 0e 0a 00 1c 01 03 09 18 1d 17 06 1a 01 03 03 11 12 13 1d 17 17 05
page 3e: 00

page 6d: 00
page 6e: 00
page 6f: 00
page 70: 12 10 08 04 08 13 15 15 00 18 15 15 10 1a 1d 12 0d 10 18 14 18 09 12 07 11 00 10 16 1c 0f 1b 06
page 71: 00
page 72: 0e 15 0c 1e 18 11 04 0d 00 12 1b 05 0f 0e 0c 16 1d 15 0e 1d 0a 17 14 17 1a 06 13 0c 14 1e 13 00
page 73: 00
page 74: 00
page 75: 19 02 16 10 10 08 17 19 17 09 16 02 15 00 00 03 14 1e 0e 13 12 1a 1e 19 0a 02 1b 08 17 1a 05 06
page 76: 00
page 77: 00
page 78: 03 11 02 14 0d 04 09 14 0e 1d 0b 0a 1c 12 03 18 0b 1d 13 18 1b 0f 1d 00 0a 19 00 14 1e 16 1a 02
page 79: 61 7f 8d 33 5c e4 7d 04 7f 7f 7f 15 e2 20 2a 0d 7f 7f 4a 27 3d 3c 7f 26 7f 7f 7f 19 6a 18 7f 7f
page 7a: f9 87 8a ea de d5 a7 cd 7f
page 7b: 04 17 16 0e 10 0f 1c 0f 19 0a 1b 1b 0e 11 1c 16 0f 06 0a 15 05 1c 08 1c 09 1d 15 0f 10 14 12 09
page 7c: 00
page 7d: 00
page 7e: 00
page 7f: 00

disk 00: 1d 1d 06 08 05 00 0e 10 1b 0e 05 0d 04 01 0c 19 02 1e 1a 1e 12 09 17 04 10 19 10 17 05 1a 04 0a
disk 01: 00
disk 02: 03 13 06 1a 18 07 10 15 04 15 01 00 12 0c 13 04 1b 0c 18 1b 08 06 0c 10 1e 0d 05 18 1b 1c 1b 09
disk 03: 0d 03 1d 10 05 16 0f 11 06 1d 1b 1b 19 1d 07 18 0c 06 16 0f 07 10 0c 12 17 17 08 0b 07 1d 1e 07
disk 04: 18 1a 0b 0e 11 09 0a 0b 08 15 11 03 03 17 0e 1a 16 12 07 12 0e 17 07 1e 1d 1e 0b 06 0d 17 14 00
disk 05: 0d 0a 11 04 10 16 0c 02 12 09 1c 09 05 0f 0b 02 12 09 1c 04 03 0f 17 1d 15 1c 0f 17 0c 0f 17 01
disk 06: 19 11 13 0a 01 10 0f 05 19 01 1a 1c 03 1d 1d 0a 0d 19 01 19 00 13 01 0e 15 10 0b 08 02 08 06 07
disk 07: 1e 17 12 04 14 10 16 01 14 1d 18 14 06 13 15 07 1c 08 01 0e 1c 15 08 0c 1b 1d 0d 0f 08 14 1b 0a
disk 08: 01 1b 18 10 1d 19 16 0b 16 0a 0a 05 00 18 18 1a 0c 15 12 08 0d 03 19 15 01 15 15 01 1e 1e 0f 0b
disk 09: 0b 0d 08 1d 09 02 01 01 0d 00 0d 1c 08 15 10 05 0d 00 0d 03 03 0e 0b 05 04 08 1d 1d 09 00 0e 1b
disk 0a: 15 17 07 04 09 16 03 15 1d 08 07 12 1a 06 01 0d 0c 03 18 01 14 08 03 0b 0c 0a 13 0e 09 06 12 1a
disk 0b: 1e 16 11 19 0b 0a 08 04 17 11 14 18 0d 0e 0f 07 1b 17 08 07 1a 08 0d 06 0f 04 09 02 13 13 08 1b
disk 0c: 0a 03 0c 16 08 0f 01 1b 1e 11 07 0e 0d 00 02 03 16 1a 0c 0a 02 0f 14 08 1b 1d 07 1b 14 0c 0d 0a
disk 0d: 11 16 09 00 01 0a 0b 1c 15 1b 17 15 17 06 17 12 13 0f 0d 1a 18 05 1a 10 01 18 1b 0d 0d 08 06 10
disk 0e: 13 14 06 09 14 1d 03 16 1a 16 1e 07 05 1e 0e 15 10 11 00 02 02 1a 02 0b 04 13 14 18 16 17 05 0d
disk 0f: 00 1a 01 02 0b 00 1c 00 02 0f 0c 17 1a 00 1d 02 0b 07 05 02 18 0a 0a 1b 0c 04 04 0b 01 14 0b 15
disk 10: 10 17 15 1d 12 0b 1a 1e 1e 03 14 04 08 18 1b 12 09 0f 09 08 06 1d 1c 05 1a 17 16 07 10 19 15 1c
disk 11: 00
disk 12: 00
disk 13: 04 06 1d 12 0f 18 14 1e 0a 10 1c 09 1c 0b 0e 0c 03 0a 15 1a 11 01 12 14 13 08 01 10 07 19 1a 02
disk 14: 00 1a 1d 11 1e 1b 0c 12 09 17 1a 0d 0f 11 10 0d 10 0b 0c 05 1d 0c 14 0b 06 1c 14 19 11 06 02 1a
disk 15: 06 08 01 0c 0f 02 0a 14 18 17 01 0b 0a 08 1e 11 13 08 0a 1b 14 19 11 01 1d 04 06 0c 13 12 0f 15
disk 16: 14 10 1c 14 02 1d 0b 19 1e 0e 18 18 01 13 18 19 00 10 0d 1c 02 0a 0d 10 07 06 16 1c 0a 12 0c 06
disk 17: 00
disk 18: 0c 03 16 09 18 11 16 16 0b 15 01 1a 16 1c 0a 05 16 05 1b 0a 1a 02 08 19 02 16 0c 11 05 07 1b 16
disk 19: 08 10 09 1d 02 0f 13 18 0e 13 04 08 1a 00 10 1c 1c 05 1e 0e 04 19 10 1e 03 1a 09 01 04 18 1c 16

disk 1a: 15 18 0a 0b 0e 04 1c 01 15 19 17 01 06 00 18 14 1d 03 19 0a 14 1e 08 16 03 00 0c 04 17 11 07 03
disk 1b: 18 03 02 14 1c 02 19 0f 14 19 03 01 1c 04 14 0c 05 02 0c 0d 03 10 16 00 09 16 1a 15 0d 0e 0c 04
disk 1c: 00
disk 1d: 00
disk 1e: 12 02 1c 1c 15 02 08 04 0c 1a 02 02 0b 0f 06 0e 18 0e 0c 16 15 0d 1a 17 16 11 05 16 17 13 12 10
disk 1f: 14 0a 13 02 17 0c 17 1b 03 1a 1d 07 14 1a 01 14 09 0e 05 0c 1b 08 0c 04 0b 06 0b 0c 10 13 13 00
disk 20: 12 1b 11 10 0b 0f 16 1a 0a 14 0a 07 01 0c 12 0f 0d 17 0e 15 11 1d 0a 0d 17 09 18 0a 04 10 11 09
disk 21: 0b 1b 0a 07 02 16 1c 0d 04 17 1a 0c 0e 0b 10 14 1c 0a 1d 13 0e 1a 07 10 1c 19 1a 10 13 18 07 1d
disk 22: 06 18 03 0a 01 09 03 19 06 0b 1a 02 15 00 1b 05 1a 0a 15 16 0c 06 0f 12 01 02 18 01 06 06 17 00
disk 23: 0c 19 16 1b 00 1e 0c 1d 17 0f 0e 1b 1b 0c 1d 1e 14 04 0b 0e 00 12 09 1c 09 01 1e 19 13 0e 0e 10
disk 24: 00
disk 25: 19 0f 15 1c 15 1d 1a 06 13 00 13 1b 16 1b 11 18 07 08 02 04 18 03 17 08 1a 1e 01 11 14 0e 00 0f
disk 26: 1d 09 07 0b 17 0b 1c 0a 1a 1e 04 12 10 18 0f 01 17 14 16 02 1e 05 08 0e 0f 0a 1a 0c 18 1a 08 03
disk 27: 17 0f 06 16 0a 18 05 00 18 04 1a 12 0d 01 18 00 13 0a 13 02 10 0d 01 00 06 11 00 03 04 12 1d 05
disk 28: 01 00 16 00 1e 00 0a 11 1b 06 07 1e 15 03 17 03 0d 15 08 1a 01 04 14 1e 13 14 14 0b 0b 15 02 1e
disk 29: 00
disk 2a: 0b 0c 1e 1e 19 14 18 1a 06 13 1e 05 0d 0b 1a 0e 04 0b 14 10 00 19 08 01 13 14 12 02 00 16 0f 00
disk 2b: 0c 1a 1c 0a 14 0a 12 09 1b 01 0e 0b 17 03 02 0d 15 02 11 0b 16 1e 03 1c 10 16 07 07 1b 16 1c 11
disk 2c: 14 06 0e 0f 02 02 00 03 0d 1e 15 11 13 19 1c 00 16 03 18 07 13 17 1a 17 1b 15 16 0a 07 0c 06 03
disk 2d: 12 10 0f 06 0b 1b 07 1b 16 04 10 0e 12 1c 04 1e 1e 1b 1d 09 07 0f 0d 1e 15 14 1b 1d 0d 0a 0d 07
disk 2e: 0a 0a 0e 1c 11 11 09 0a 0e 18 1b 19 06 1c 09 0a 0e 03 10 1d 03 1d 0c 07 10 02 0a 17 09 16 08 17
disk 2f: 00
disk 30: 06 14 1e 05 11 01 12 0d 0b 1c 0e 0d 18 1c 05 04 1e 15 17 1e 1d 12 06 15 0c 14 0e 0e 11 16 0c 17
disk 31: 00
disk 32: 18 16 1c 16 14 13 07 17 19 07 0b 0d 16 02 03 1c 17 0c 05 00 04 1b 02 1a 05 18 07 15 17 10 07 0f
disk 33: 07 02 07 18 1b 18 1b 1c 10 0b 04 1a 0c 13 06 0c 00 0c 03 0e 1e 14 0a 02 03 14 00 1e 08 13 06 16
disk 34: 00
disk 35: 1b 03 02 0f 10 00 0d 04 05 17 00 06 0f 19 18 0f 0f 19 10 18 0d 03 1c 06 11 14 0f 0b 01 0d 02 10
disk 36: 09 0b 02 04 07 1c 1c 08 1d 04 14 03 11 1e 16 15 14 19 0f 1b 16 16 07 18 1d 01 0d 15 19 19 09 0e
disk 37: 19 10 09 1c 01 13 17 19 01 13 09 14 01 0e 0f 08 08 16 11 0b 0e 13 12 15 1c 1e 04 0e 11 03 08 05
disk 38: 00 15 13 09 12 0d 00 07 1e 0a 03 1d 19 06 00 06 15 1d 0a 01 06 1b 03 04 1e 0c 04 14 1e 04 1d 17
disk 39: 0d 0e 1a 02 1d 1c 0d 1e 18 15 10 16 11 18 07 0d 08 00 15 12 07 19 16 17 12 09 14 07 10 19 0f 1a
disk 3a: 11 09 0d 1b 1b 19 1d 1e 00 08 02 19 04 18 05 06 0a 1b 07 05 18 08 11 0e 15 0f 07 04 1a 07 07 11
disk 3b: 0c 08 0e 02 19 07 02 04 0a 0a 13 0a 06 0e 0f 0f 0b 10 1e 0b 15 0d 16 0c 18 1e 16 17 01 15 15 04
disk 3c: 15 06 0c 16 08 13 01 15 07 0d 1e 18 05 18 0c 0f 08 05 1e 0c 06 18 03 0c 03 01 1e 0f 14 04 0b 01
disk 3d: 06 13 11 0c 12 03 1c 0a 0a 1b 12 1e 18 17 1d 15 01 11 0e 10 1e 1e 16 19 00 0d 0e 07 1c 00 1a 1e
disk 3e: 12 1d 00 08 1b 01 15 09 07 01 08 0b 14 19 12 04 10 1c 07 15 06 09 10 19 10 05 11 01 16 11 0a 13
disk 3f: 15 16 13 05 1d 01 06 03 14 0e 1e 05 02 00 09 19 0b 10 09 08 16 00 11 1d 09 0f 13 08 05 1a 00 07
disk 40: 07 15 0f 1e 06 0e 17 1b 09 18 07 06 14 1e 19 01 0d 17 17 19 00 02 0f 07 13 15 0d 00 0c 11 0f 04
disk 41: 00
disk 42: 04 10 00 05 0a 0e 0b 08 1d 03 04 17 09 1a 1a 0f 13 0b 17 1e 13 0b 1c 00 17 1d 07 13 03 0c 1d 02
disk 43: 12 08 00 05 1c 18 03 18 15 09 12 08 1a 00 04 0f 10 10 16 0e 09 1d 03 0c 0f 1a 02 1e 0e 00 09 0f
disk 44: 00 12 0b 12 1e 04 09 10 16 05 1a 0b 00 1e 0b 04 12 17 0a 08 12 15 10 14 17 05 1d 17 10 02 09 12
disk 45: 00
disk 46: 00
disk 47: 08 10 1a 19 13 16 00 04 00 0d 0f 00 19 10 16 1d 17 17 13 10 17 18 09 18 04 10 1c 06 1b 0f 16 04

disk 48: 0d 0c 02 15 0f 06 17 00 1b 10 19 0f 04 0e 06 11 00 15 18 11 0a 14 0c 1d 1c 0d 18 07 1c 14 00 13
disk 49: 0e 06 03 0f 0a 1a 03 02 0a 10 16 13 07 02 14 03 18 13 1b 0d 0e 13 03 1a 12 15 02 08 04 0a 0b 1c
disk 4a: 1c 10 03 15 1d 14 06 12 13 13 09 11 0d 16 04 16 1c 0d 08 07 0a 13 10 05 0e 07 06 0b 15 0d 07 00
disk 4b: 18 05 1d 1a 05 1c 08 1a 06 1a 1a 11 15 17 13 0a 17 1c 10 1a 0b 0d 15 09 18 18 06 0c 1b 0f 01 19
disk 4c: 0f 12 16 05 12 12 1e 0c 06 07 15 18 09 19 1c 0b 02 1c 07 1e 0c 0f 0d 02 0b 02 0f 03 12 10 1c 19
disk 4d: 05 04 17 02 07 04 0b 05 05 14 09 09 1b 10 1d 04 13 0a 00 04 00 12 01 1e 03 11 0b 1c 02 10 02 09
disk 4e: 19 06 0f 0e 02 18 06 16 07 17 1d 02 07 16 07 07 0c 15 1a 13 05 0e 01 14 16 06 00 14 18 1c 00 1b
disk 4f: 18 0c 15 1e 08 13 1e 09 19 02 14 10 17 10 15 03 1a 10 15 17 0f 1a 16 07 02 16 1b 0a 09 14 1d 13
disk 50: 11 07 1a 0d 02 08 19 0d 16 06 0e 0c 0f 15 1a 0e 00 1b 1c 17 0c 1e 1b 1c 0c 12 05 06 0e 09 03 00
disk 51: 07 09 14 03 12 10 08 11 04 0e 0b 05 0c 1c 19 0f 18 16 0f 06 08 0c 03 12 0f 14 00 1c 14 11 09 16
disk 52: 03 0a 1d 11 0f 14 03 04 0c 0b 0e 11 02 08 19 17 13 01 14 02 06 07 1e 0a 09 03 0c 18 0a 0d 03 04
disk 53: 14 06 17 0d 0c 05 12 15 18 19 0a 10 13 0c 00 16 1b 0e 1c 01 12 0c 13 0b 16 06 08 0c 0c 0c 1d 0f
disk 54: 0e 07 18 10 04 1d 1b 02 0b 14 19 1b 0c 02 19 11 01 0c 18 13 14 1d 17 1a 0e 16 14 1d 1b 19 06 08
disk 55: 10 0f 13 00 03 11 14 04 10 08 1d 14 19 18 18 1c 05 02 0e 09 09 03 1c 05 13 02 00 1d 0a 14 03 07
disk 56: 0c 1b 0f 18 14 1c 0b 0a 1a 13 0d 19 06 06 11 18 0c 00 12 1b 12 00 0a 1e 09 15 04 13 0f 01 1e 06
disk 57: 06 03 1a 1e 06 1b 02 1d 03 05 16 11 0e 18 1d 19 06 19 1d 17 13 1e 0d 1c 18 17 0e 0a 19 0a 03 06
disk 58: 0e 02 03 0e 19 02 1c 00 1d 09 0d 0f 0c 08 0e 0f 11 0b 05 12 01 03 0a 10 02 1d 16 0e 11 0a 00 1b
disk 59: 00
disk 5a: 00
disk 5b: 00
disk 5c: 08 15 14 0f 18 06 0f 08 15 1e 18 0f 00 06 04 07 1d 18 10 02 07 13 18 16 12 1c 0f 02 09 07 1b 19
disk 5d: 12 18 13 17 0a 10 00 0a 1d 02 09 00 17 14 0f 0f 1d 14 0a 1a 19 0e 0b 13 11 0b 0b 05 09 14 1a 04
disk 5e: 16 09 07 09 1e 18 05 07 14 0f 07 04 07 09 08 09 13 18 11 0a 05 02 1e 06 13 0a 02 14 12 01 14 1c
disk 5f: 00
disk 60: 0e 00 15 00 12 1b 10 0c 16 06 1b 00 1c 01 1b 0d 10 09 16 1d 04 10 04 18 01 0a 06 16 06 1b 18 03
disk 61: 16 06 02 0c 18 09 12 15 05 11 04 18 0e 1b 0d 0c 0d 0d 18 09 00 14 04 04 0a 0e 07 1c 09 1c 08 00
disk 62: 1b 06 01 19 13 1a 1a 03 08 1a 11 18 1c 00 07 02 07 0c 1c 06 0f 0d 1d 14 0c 02 08 03 1e 09 05 00
disk 63: 0d 1d 10 0c 03 16 0d 17 1d 09 0c 16 1b 1c 0b 07 13 19 1a 07 0c 10 07 02 10 07 11 09 0f 1c 03 09
disk 64: 05 07 15 13 0a 19 0d 08 0e 06 17 1d 16 11 02 1d 1b 19 16 02 09 0f 12 06 1b 0f 16 1b 07 06 11 03
disk 65: 0c 19 13 19 1b 14 1b 00 1c 0f 0f 07 09 11 0b 00 0a 04 1e 02 16 17 1e 16 16 13 1b 1a 0f 1d 11 0c
disk 66: 1b 0b 12 01 01 12 08 0b 04 1e 18 1a 0a 03 04 1c 0e 06 1c 1b 06 18 05 03 06 14 0e 0b 17 06 17 16
disk 67: 0b 02 02 03 0c 0e 1c 18 1c 05 01 11 0d 13 1a 16 14 01 17 01 1a 09 09 0c 1d 09 17 1e 03 10 02 03
disk 68: 1a 03 02 0f 0d 18 02 17 1c 18 0a 13 0c 06 06 09 08 12 10 0e 14 00 13 02 1e 12 0f 04 15 1e 0a 14
disk 69: 1c 0c 05 06 00 0f 1e 0b 10 04 17 0b 08 19 15 11 16 04 15 11 13 0a 0d 02 0e 0e 11 0f 12 11 1e 04
disk 6a: 1a 0e 11 0c 15 06 0a 16 0a 1e 0e 00 0f 1a 05 0d 11 1b 00 16 02 15 1d 13 00 08 0e 1b 0d 07 17 16
disk 6b: 08 02 19 07 15 19 14 01 07 05 0d 1b 09 00 1e 13 08 16 0b 1a 19 16 07 19 16 0f 0b 1b 12 0b 15 06
disk 6c: 1c 08 07 16 0a 12 0c 1c 13 1a 10 11 18 1a 17 0d 13 16 06 03 13 08 16 00 15 01 00 05 0a 03 12 1a
disk 6d: 0e 19 09 1e 0b 1b 18 12 0b 07 0d 1a 14 14 0b 09 16 04 03 11 1c 00 04 11 0b 18 11 11 01 0a 08 0e
disk 6e: 14 15 0a 00 07 1c 0c 0e 09 00 11 02 19 11 03 06 01 05 09 0b 0c 01 1d 0c 1e 05 14 1a 0c 17 08 15
disk 6f: 00
disk 70: 00
disk 71: 00
disk 72: 1e 0c 1d 12 00 01 05 1e 05 1b 16 1c 13 07 18 00 0a 0a 18 0d 04 04 05 0c 10 00 1e 12 08 12 1e 04
disk 73: 17 17 00 07 03 1e 06 04 16 18 16 10 1b 12 01 19 00 1e 11 13 07 1a 0b 11 01 14 1a 13 19 11 11 19
disk 74: 15 00 0d 01 1b 14 17 12 03 10 19 06 17 03 08 0e 01 0b 10 10 1b 18 07 1b 11 18 03 0c 0a 0f 1b 02
disk 75: 03 08 17 07 0c 05 1e 18 0b 1e 1b 09 02 16 13 08 00 08 1c 15 14 15 08 1b 0e 1c 12 15 1b 1e 16 1d

```
disk 76: 0f 15 13 1e 01 10 1a 0c 01 0c 12 1d 05 07 08 05 0c 1a 0f 08 1d 0e 05 17 04 14 01 11 17 08 0b 16
disk 77: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
disk 78: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
disk 79: 08 17 16 03 0c 1b 13 01 02 0f 1c 18 1d 08 06 18 0d 1c 15 09 1d 01 1d 1c 12 04 06 01 0f 01 19 0f
disk 7a: 16 1a 0e 18 0a 05 09 07 08 1d 02 17 08 0c 1e 12 13 19 01 0a 03 1a 15 1a 11 02 0e 1a 1d 1a 01 1c
disk 7b: 10 02 11 18 1c 03 16 03 13 10 00 05 0c 15 1e 08 08 15 0e 12 1a 17 0d 0b 15 12 14 0f 16 07 19 0c
disk 7c: 0c 06 06 0e 11 15 0e 1d 0c 00 10 14 07 17 06 1c 0d 1d 13 16 06 1b 1e 04 11 04 1b 1a 1c 17 0c 13
disk 7d: 06 03 13 16 1b 14 05 0c 11 17 1a 1c 1c 14 14 0d 15 0b 1b 09 10 0a 10 14 07 0e 11 15 12 16 03 0d
disk 7e: 0d 08 02 08 10 08 0b 16 0f 02 02 1a 05 0c 15 1e 10 17 13 18 0e 06 10 13 04 12 0e 0b 0f 0b 0b 1e
disk 7f: 0f 16 0d 11 08 05 19 15 09 0c 11 02 13 1e 15 0e 03 10 12 15 1b 08 03 1e 10 1b 12 1b 12 16 13 1d
-----
```

六、（10 分）给出下面程序 `fork-example.cpp` 的输出结果；

```
=====fork-example.cpp=====
```

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

int globalVariable = 20;

int main(void)
{
    char sIdentifier[20];
    int iStackVariable = 20;

    pid_t pID = fork();
    if (pID == 0) // child
    {
        strcpy(sIdentifier, "Child Process: ");
        globalVariable++;
        iStackVariable--;
    }
    else if (pID < 0) // failed to fork
    {
        printf("Failed to fork\n");
        exit(1);
        // Throw exception
    }
    else // parent
```

```
{  
    strcpy(sIdentifier, "Parent Process:");  
}  
  
globalVariable++;  
iStackVariable--;  
printf("%s",sIdentifier);  
printf(" Global variable: %d", globalVariable);  
printf(" Stack variable: %d\n", iStackVariable);  
}
```

=====