北京大学信息科学技术学院考试试卷

考试科目：计算机系统导论姓名：学号：

考试时间：2016年1月4日小班教师:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 总分 |
| 分数 |  |  |  |  |  |  |  |  |  |
| 阅卷人 |  |  |  |  |  |  |  |  |  |

**北京大学考场纪律**

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过15分钟不得入场。在考试开始30分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

装订线内不要答题

以下以下为答题纸，共页

以下为试题和答题纸，共页。

|  |
| --- |
| 得分 |
|  |

## 第一题单项选择题（每小题1分，共20分）

注：请将选择题答案填写在下表中

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 答案 |  |  |  |  |  |  |  |  |  |  |

1. **下面哪条指令不会引起esp的变化？**
   1. **movl %esp, %ebp**
   2. **pushl %ebp**
   3. **call printf**
   4. **subl $20, %esp**

答案：A，考察汇编指令和栈的理解。

1. **单精度浮点数27.5实际存储在内存中的十六进制数值为：**
   1. **0x41ee0000**
   2. **0x425c0000**
   3. **0x41dc0000**
   4. **0x025c0000**

参考信息：单精度浮点数阶码8位，尾数23位

**答案：C，考察浮点数的理解。**

1. **下面哪条指令不是X86正确的寻址方式**
2. **movl $34, (%eax)**
3. **movl (%eax), %eax**
4. **movl $23, 10(%edx, %eax)**
5. **movl (%eax), 8(%ebx)**

**答案：D。考察寻址方式的理解。**

**以下关于静态库链接的描述中，正确的是：**

1. **链接时，链接器会拷贝静态库中的所有目标模块。**
2. **使用库的时候必须把它们放在命令行的结尾处。**
3. **如果库不是相互独立的，那么它们必须排序。**
4. **每个库在命令行只须出现一次即可。**

【答案】C。

解释：A: 链接时只需拷贝用到的目标模块；B：静态库也可以放在目标文件的前面；C: 正确；D：如果相互调用的库，在命令行必须重复出现。

1. **在foo.c文件中包含如下代码：**

int foo(void) {

int error = printf("You ran into a problem!\n");

return error;

}

经过编译和链接之后，字符串"You ran into a problem!\n"会出现在哪个段中？

1. .bss
2. .data
3. .rodata
4. .text

【答案】C.

1. **一段程序中阻塞了SIGCHLD和SIGUSR1信号。接下来，向它按顺序发送SIGCHLD，SIGUSR1，SIGCHLD信号，当程序取消阻塞继续执行时，将处理这三个信号中的哪几个？**
   1. **都不处理**
   2. **处理一次SIGCHLD**
   3. **处理一次SIGCHLD，一次SIGUSR1**
   4. **处理所有三个信号**

**答案：C。**

1. **学完本课程后，几位同学聚在一起讨论有关异常的话题，请问你认为他们中谁学习的结果有错误？**
   1. **发生异常和异常处理意味着控制流的突变。**
   2. **与异常相关的处理是由硬件和操作系统共同完成的。**
   3. **异常是由于计算机系统发生了不可恢复的错误导致的。**
   4. **异常的发生可能是异步的，也可能是同步的。**

答案：C。异常不一定是不可恢复的，也可能是可恢复的、甚至是有意产生的。

1. 下列说法正确的是：

A．SIGTSTP信号既不能被捕获，也不能被忽略

B．存在信号的默认处理行为是进程停止直到被SIGCONT信号重启

C．系统调用不能被中断，因为那是操作系统的工作

D．子进程能给父进程发送信号，但不能发送给兄弟进程

答案：B

A：SIGTSTP可以被忽略；既不能被捕获又不能被忽略的是SIGKILL和SIGSTOP

C：系统调用可以被中断，如read这样的慢速系统调用

D：可以通过kill(0, …)给整个组的进程发送信号

1. 在系统调用成功的情况下，下面哪个输出是可能的？

int main() {

intpid = fork();

if (pid == 0) {

printf("A");

} else {

pid = fork();

if (pid == 0) {

printf("A");

} else {

printf("B");

}

}

exit(0);

}

1. AAB
2. AAA
3. AABB
4. AA

答案：A

总共产生3个子进程，共输出2个A和1个B

1. **以下四句都是关于Unix I/O的说法。其中正确的是：**
2. 从网络套接字（socket）读取内容时，可以通过反复读的方式处理不足值问题，直到读完所需要的数量或遇到EOF为止。
3. 以O\_RDWR方式打开文件后，文件会有两个指针，分别记录读文件的当前位置和写文件的当前位置。
4. 用read函数直接读取控制台输入的文本行，会自动在行末追加‘\0’字符。
5. 使用dup2(4, 1)成功进行重定向后执行close(4)，会导致1号文件描述符也不可用。

参考信息：O\_RDWR表示文件可读可写；dup2(oldfd, newfd)表示将oldfd重定向给newfd。

**答案：A。**

**说明：B. 并没有两个指针，读写操作是共用一个指针的；C. read操作不会追加‘\0’字符；D. close(4)后1号描述符仍然可用。**

1. **下面是一段C程序代码：**

#include <stdio.h>

#include "csapp.h"

int main()

{

printf("2");

if (Fork())

{

printf("33");

Write(STDOUT\_FILENO, "lol", 3);

}

else

{

Sleep(1);

printf("233");

Write(STDOUT\_FILENO, "hhhh", 4);

}

fflush(stdout);

return 0;

}

**编译后运行程序，程序正常退出。那么程序的输出是：**

* 1. **233lol233hhhh**
  2. **lol233hhhh2233**
  3. **233lol2233hhhh**
  4. **2lol33hhhh233**

**答案：B。**

**说明：printf所属的标准IO是有缓冲区的，直到关闭文件、遇到换行符或者fflush时才会输出；write则没有缓冲区立刻输出。**

1. **根据编译器安全优化的策略，如下手工程序代码的优化，哪个达不到优化效果？**
   1. **循环展开，以减少循环的迭代次数**
   2. **消除不必要的存储器引用，减少访存开销**
   3. **将函数调用移到循环内，以提高程序的模块性**
   4. **分离多个累计变量，以提高并行性**

答案：C

1. 动态管理器分配策略中，最适合“最佳适配算法”的空白区组织方式是：
2. **按大小递减顺序排列**
3. **按大小递增顺序排列**
4. **按地址由小到大排列**
5. **按地址由大到小排列**

**答案：B。**

**说明：最佳适应算法的空白区是按小大递增顺利链接在一起。**

1. 虚拟**内存**管理方式可行性的基础是：
2. **程序执行的离散性**
3. **程序执行的顺序性**
4. **程序执行的局部性**
5. **程序执行的并发性**

**答案： C。考察虚拟内存管理方式可行性是利用了程序执行的局部性。**

1. Intel的IA32体系结构采用二级页表，称第一级页表为页目录（Page Directory），第二级页表为页表（Page Table）。页面的大小为4KB，页表项4字节。以下给出了页目录与若干页表中的部分内容，例如，页目录中的第1个项索引到的是页表3，页表1中的第3个项索引到的是物理地址中的第5个页。则十六进制逻辑地址8052CB经过地址转换后形成的物理地址应为十进制的（）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 页目录 | | 页表1 | | 页表2 | | 页表3 | |
| VPN | 页表号 | VPN | 页号 | VPN | 页号 | VPN | 页号 |
| 1 | 3 | 3 | 5 | 2 | 1 | 2 | 9 |
| 2 | 1 | 4 | 2 | 4 | 4 | 3 | 8 |
| 3 | 2 | 5 | 7 | 8 | 6 | 5 | 3 |

* 1. **21195**
  2. **29387**
  3. **21126**
  4. **47195**

答案：4KB=2^12，所以页内地址有12位。4KB/4B=1K，所以页目录和每个页表中的页表项数为1K个。因此，在32位的虚拟地址中，最低的12位为页内地址（Offset），最高的10位为页目录的虚拟地址（Dir），中间10位为页表的虚拟地址（Table）。十六进制逻辑地址8052CB转换为二进制后为10 0000000101 001011001011，Dir为10，即10进制的2，在表中对应到页表1。Table为101，即10进制的5，在表中对应到物理页面7。因此，物理地址应为7的二进制111和Offset的拼合，即111001011001011，转换为十进制为29387，答案为B。

1. 已知某系统页面长8KB，页表项4字节，采用多层分页策略映射64位虚拟地址空间。若限定最高层页表占1页，则它可以采用多少层的分页策略？
   1. **3层**
   2. **4层**
   3. **5层**
   4. **6层**

答案：C。由题意，64位虚拟地址的虚拟空间大小为264.页面长为8KB,页表项4字节，所以一个页面可存放2K个表项。由于最高层页表占1页，也就是说其页表项个数最多为2K个，每一项对应一页，每页又可存放2K个页表项，依次类推可知，采用的分页层数为：5层。

1. HTTP协议中，哪个命令可以用来获取动态内容？
   1. **HEAD**
   2. **GET**
   3. **POST**
   4. **PUT**

答案：B或C都正确

说明：POST只能用于获得静态内容

1. 下列关于计算机网络概念的说法中，哪一项是正确的？
2. **HUB会把它任意端口上接收到的帧只转发到它的目的地去**
3. **当在不同的LAN中的主机A和主机B通信的过程中，他们的数据包中的LAN frame header不会变化**
4. **162.105.0.0是一个B类地址**
5. **同一台主机每次进入相同的网络，通过动态地址分配的到的IP地址总是相同的**

答案：C

说明：HUB会广播它任意端口上接收到的帧；LAN frame header的目的地地址在经历的不同的LAN中会不断调整；动态分配地址和主机地址没有唯一映射关系

1. 有如下代码：

int counter = 0;

void \* thread(void \* vargp)

{

intthread\_var = ((int \*) vargp);

staticintthread\_counter = 0;

thread\_internal(thread\_var);

thread\_counter ++;

return NULL;

}

int main (intargc, const char \*\* argv)

{

int tid1, tid2;

intvar = atoi(argv[1]);

Pthread\_create(&tid1, NULL, thread, (void \*)var);

Pthread\_create(&tid2, NULL, thread, (void \*)var);

Pthread\_join(tid1, NULL);

Pthread\_join(tid2, NULL);

return 0;

}

则，线程 tid1 与线程 tid2 可以共享的变量是

1. **counter, var**
2. **counter, thread\_counter**
3. **var, thread\_counter**
4. **thread\_var, thread\_counter**

答案：B

（本题考查对线程中共享变量的概念的理解。因为counter是全局变量；thread\_counter是静态局部变量，所以两个线程可以共享它们）

1. 有四个信号量，初值分别为：a=1，b=1，c=1，d=1。

|  |  |  |
| --- | --- | --- |
| 线程① | 线程② | 线程③ |
| P(a);  P(d);  P(c);  P(b);  V(c);  V(b);  V(d);  V(a); | P(d);  P(a);  P(c);  P(b);  V(d);  P(d);  V(a);  V(b);  V(c);  V(d); | P(d);  P(c);  P(b);  P(a);  V(c);  V(b);  V(a);  V(d); |

**下列哪两个线程并发执行时，一定不会发生死锁？**

1. **①, ②**
2. **①, ③**
3. **②, ③**
4. **以上选项均不正确**

答案：D

（本题考查对死锁概念的理解，本题的情况是任意两个线程并发执行，都会产生死锁）

|  |
| --- |
| 得分 |
|  |

## 第二题（12分）汇编

下面分别是一个程序的C语言代码、汇编语言代码，以及其执行结果，请根据其逻辑分别填写空出来的内容：

1. C语言代码

#include <stdio.h>

long f1(long x, long y)

{

return (1) ;

}

long f2(long x, long y)

{

return (2) ;

}

long a[6] = {1, 0, 0, 0, 0, 0};

void foo(void)

{

long (\*f)(long, long);

(3) long count = 0;

longi;

for (i=0; i< (4) ; i++) {

if ((count % 2) == 0)

f = f1;

else

f = f2;

a[i+1] = f(a[i], (5) );

count ++;

}

for (i=0; i<6; i++) {

printf("%ld\n", a[i]);

}

}

int main()

{

foo();

foo();

}

1. 汇编语言代码

.LC0:

.string (6)

f1:

leaq (%rdi,%rsi), %rax

ret

f2:

movq %rdi, %rax

imulq %rsi, %rax

ret

foo:

pushq %r12

pushq %rbp

(7)

movl $a, %ebx

movl $a + (8) , %r12d

movl $f1, %ebp

.L5:

movq count, %rsi

movq %rsi, %rax

andl $1, %eax

movl $f2, %eax

cmove %rbp, %rax

movq (%rbx), %rdi

call \*%rax

movq % (9) , 8(%rbx)

addq $1, count

addq $8, %rbx

cmpq %r12, %rbx

jne .L5

movl $0, %ebx

.L6:

movq a(,%rbx,8), %rdx

movl $.LC0, %esi

movl $1, %edi

movl $0, %eax

call printf

addq $1, %rbx

cmpq $6, %rbx

jne .L6

popq %rbx

popq %rbp

(10)

ret

1. 输出结果

1

1

1

3

(11)

13

1

5

11

77

85

(12)

答案：（每空一分）

1) (x + y)

2) (x \* y)

3)static

4) 5

5) count

6) "%ld\n"（必需有引号）

7)pushq %rbx

8) 0x40或者64（40也可给分，因为汇编默认是16进制）

9) rax

10)popq %r12

11) 9

12) 765

|  |
| --- |
| 得分 |
|  |

## 第三题（13分）处理器

如图所示，每个模块表示一个单独的组合逻辑单元，每个单元的延迟已在图中标出。通过在两个单元间添加寄存器的方式，可以对该数据通路进行流水化改造。假设每个寄存器的延迟为20ps。设计人员考虑在额外增加一个模块F支持新的指令功能，形成图形状的流水线。提示：每个模块必须在一个时钟周期内。



1）如果没有F模块，请计算该流水线改造前的吞吐率，并说明计算过程。结果保留小数点后两位。

1000/(50+20+30+40+60+20) = 1000/220 = 4.55GIPS（过程和结果2分）

2）如果有F模块，请计算该流水线改造前的吞吐率，并说明计算过程。结果保留小数点后两位。

1000/(50+70+40+60+20) = 1000/240 = 4.17GIPS（过程和结果2分）

3）如果有F模块，改造为一个二级流水线（可以插入多个寄存器），为获得最大的吞吐率，寄存器应在哪里插入？请计算该流水线的吞吐率，并说明计算过程。结果保留小数点后两位。

插入在FD间和CD间（2分）

1000/(50+70+20) = 1000/140 = 7.14 GIPS（过程和结果2分）

4）如果有F模块，改造为一个三级流水线（插入多个寄存器），为获得最大的吞吐率，寄存器应在哪里插入？请计算该流水线的吞吐率，并说明计算过程。结果保留小数点后两位。

插入在AF间、FD间、CD间（3分，可以酌情部分给分）

1000/(40+60+20) = 1000/120 = 8.33 GIPS（过程和结果2分）

|  |
| --- |
| 得分 |
|  |

## 第四题（10分）链接

在x86\_64环境下，考虑如下2个文件：main.c和foo.c：

/\* main.c \*/

#include <stdio.h>

longlong \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

const char\* foo(int);

int main(intargc, char \*\*argv){

int n = 0;

sscanf(argv[1], "%d", &n);

printf(foo(n));

printf("%llx\n", a);

}

/\* foo.c \*/

#include <stdio.h>

int a[2];

static void swapper(intnum){

int swapper;

if (num % 2){

swapper = a[0];

a[0] = a[1];

a[1] = swapper;

}

}

const char\* foo(intnum){

static char out\_buf[50];

swapper(num);

sprintf(out\_buf, "%x\n", \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

returnout\_buf;

}

1. 对于每个程序中的相应符号，给出它的属性（局部或全局，强符号或弱符号）（提示：如果某表项中的内容无法确定，请画X。）

main.c

|  |  |  |
| --- | --- | --- |
|  | 局部或全局？ | 强或弱？ |
| a |  |  |
| foo |  |  |

foo.c

|  |  |  |
| --- | --- | --- |
|  | 局部或全局？ | 强或弱？ |
| a |  |  |
| foo |  |  |
| out\_buf |  |  |

1. 根据如下的程序运行结果，补全程序【在程序空白处填空即可】。

$ gcc -o test main.cfoo.c

$ ./test 1

bffedead

cafebffedeadbeef

$ ./test 2

beefcafe

deadbeefcafebffe

1. 现在有一位程序员要为这个程序编写头文件。假设新的头文件名称为foo.h，内容如下：

extern long long a;

extern char \*foo(int);

然后在main.c和foo.c中分别引用该头文件，请问编译链接能通过吗？请说明理由。

答案：

1. （5分）对于每个程序中的相应符号，给出它的属性（局部或全局，强符号或弱符号）（提示：如果某表项中的内容无法确定，请画X。）

（每格0.5分）

main.c

|  |  |  |
| --- | --- | --- |
|  | 局部或全局？ | 强或弱？ |
| A | 全局 | 强 |
| foo | 全局 | 弱 |

foo.c

|  |  |  |
| --- | --- | --- |
|  | 局部或全局？ | 强或弱？ |
| A | 全局 | 弱 |
| foo | 全局 | 强 |
| out\_buf | 局部 | X |

1. （3分）根据程序运行结果，补全程序

long longa = 0xdeadbeefcafebffe; （1分）

\*(int\*)((unsigned long long)a + 2) （2分）

1. 请问编译链接能通过吗？请说明理由。

不能。无论如何声明a的类型都会造成在至少一个文件内引起声明和定义冲突。

结论1分，理由2分。（结论错不得分）

|  |
| --- |
| 得分 |
|  |

## 第五题（10分）异常

以下程序运行时系统调用全部正确执行，且每个信号都被处理到。请给出代码运行后所有可能的输出结果。

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

int c = 1;

void handler1(int sig) {

c++;

printf("%d", c);

}

int main() {

signal(SIGUSR1, handler1);

sigset\_t s;

sigemptyset(&s);

sigaddset(&s, SIGUSR1);

sigprocmask(SIG\_BLOCK, &s, 0);

intpid = fork()?fork():fork();

if (pid == 0) {

kill(getppid(), SIGUSR1);

printf("S");

sigprocmask(SIG\_UNBLOCK, &s, 0);

exit(0);

} else {

while (waitpid(-1, NULL, 0) != -1);

sigprocmask(SIG\_UNBLOCK, &s, 0);

printf("P");

}

return 0;

}

答：

答案：

共5种：

S2PS2P

SS2P2P

S2SP2P

|  |
| --- |
| 得分 |
|  |

## 第六题（15分）虚拟内存地址转换

为了提升虚拟内存地址的转换效率，降低遍历两级页表结构所带来的地址转换开销，英特尔处理器中引入了大页TLB，即一个TLB项可以涵盖整个4MB对齐的地址空间（针对32位模式）。只要设置页目录页中页目录项（PDE）的大页标志位，即可让MMU识别这是一个大页PDE，并加载到大页TLB项中。大页PDE中记录的物理内存页面号必须是4MB对齐的，并且整个连续的4MB内存均可统一通过该大页PDE进行地址转换。

在32位的Linux系统中，为了方便访问物理内存，内核将地址0~768MB间的物理内存映射到虚拟内存地址3GB~3GB+768MB上，并通过大页PDE进行进行该区间的地址转换。任何0~768MB的物理内存地址可以直接通过加3G（0xC0000000）的方式得到其虚拟内存地址。在内核中，除了该区间的内存外，其他地址的内存通常都通过普通的两级页表结构来进行地址转换。

假设在我们使用的处理器中有2个大页TLB项，其当前状态如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 索引号 | TLB标记 | 页面号 | 有效位 |
| 0 | 0xC4812 | 0x04812 | 1 |
| 1 | 0xC9C33 | 0x09C33 | 1 |

有4个普通TLB项，当前的状态如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 索引号 | TLB标记 | 页面号 | 有效位 |
| 0 | 0xF8034 | 0x04812 | 1 |
| 1 | 0xF8033 | 0x09812 | 1 |
| 2 | 0xF4427 | 0x12137 | 1 |
| 3 | 0xF44AE | 0x17343 | 1 |

当前页活跃的目录页（PD）中的部分PDE的内容如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PDE索引 | 页面号 | 其他标志 | 大页位 | 存在位 |
| 786 | 0x04800 | … | 1 | 1 |
| 807 | 0x09C00 | … | 1 | 1 |
| 977 | 0x09C33 | … | 0 | 1 |
| 992 | 0x09078 | … | 0 | 1 |

*注：普通页面大小为4KB，并且4KB对齐。每个页面的页面号为其页面起始物理地址除以4096得到。大页由连续1024个4KB小页组成，且4MB对齐。*

1. 分析下面的指令序列，

movl $0xC4812024, %ebx

movl $128, (%ebx)

movl $0xF8034000, %ecx

movl $36(%ecx), %eax

请问，执行完上述指令后，eax寄存器中的内容是（）；在执行上述指令过程中，共发生了（）次TLB miss？同时会发生（）次page fault？

注：不能确定时填写“--”。

1. 请判断下列页面号对应的页面中，哪些一定是页表页？哪些不是？哪些不确定？

|  |  |
| --- | --- |
| 页面号 | 是否为页表页（是/不是/不确定） |
| 0x04800 | 4 |
| 0x09C33 | 5 |
| 0x09812 | 6 |

1. 下列虚拟地址中哪一个对应着够将虚拟内存地址0xF4427048映射到物理内存地址0x14321048的页表项（）？

（A）0x09C33027 （B）0xC9C3309C

（C）0xC9C33027 （D）0x09C3309C

通过上述虚拟地址，利用movl指令修改对应的页表项，完成上述映射，在此过程中，是否会产生TLB miss？（）（回答：会/不会/不确定）

修改页表项后，是否可以立即直接使用下面的指令序列将物理内存地址0x14321048开始的一个32位整数清零？为什么？

movl $0xF4427048, %ebx

movl $0, (%ebx)

答：

答案：

第1小题（各1分）

（1）128；（2）0；（3）0；

两个虚拟地址映射的是同一块物理内存；因此读出的就是写入的；此过程中全部TLB命中，因而既无TLB miss，也不会有page fault。

第2小题（各2分）

（1）不确定；因为是大页，一定不是当前页目录项对应的页表页，但不一定该页面不会用作其他页目录项对应的页表页。

（2）是；当前页目录项（977）对应的页表页。

（3）不确定；任何页面都可能用作页表页。

第3小题

B；（2分）

虚拟地址对应的页表页的页面号（0x09C33）已知，通过其地址直接加3G（即0xC0000000），即可得到当前页表页的基地址（0xC9C33000），在加上对应的第0x27乘以4到页内偏移。

不会；（2分）

因为地址0xC9C33000在大页映射范围内，已经被大页TLB项覆盖到了，会直接命中。

不能直接修改。（1分）

因为TLB项中的内容和页表中的内容不一致，需要将对应的TLB项设置为失效，然后通过TLB miss重新加载页表结构中新的地址映射关系，之后才能访问对应的虚拟地址。（1分）

|  |
| --- |
| 得分 |
|  |

## 第七题（10分）网络

1. 请根据web应用在计算机网络中的定义以及其在协议栈自上而下在软件中的实现，把以下关键字填入表格

*注：同一个关键词可能被填入多次；不是每一个关键词都必须被填入*

Streams (end to end), Datagrams, web content, IP, TCP, UDP, Kernel code, User code

|  |  |  |
| --- | --- | --- |
| 协议 | 数据包类型 | 软件实现 |
| HTTP |  |  |
|  |  |  |
|  |  |  |

答案：（每空1分，共8分）

|  |  |  |
| --- | --- | --- |
| 协议 | 数据包类型 | 软件实现 |
| HTTP | Web content | User code |
| TCP | Streams | Kernel code |
| IP | Datagrams | Kernel code |

1. 以下关于互联网的说法中哪些是正确的？并简要说明原因

A. 在client-server模型中，server通常使用监听套接字listenfd和多个client同时通信

B. 在client-server模型中，套接字是一种文件标识符

C. 准确地说，IP地址是用于标识主机的adapter (network interface card)，并非主机

D. Web是一种互联网协议

E. 域名和IP地址是一一对应的

F. Internet是一种internet

答：

答案：B, C, F（全对得2分，有错得0分，部分对得1分）

说明的原因供评分时参考

说明：server会为每一个client单独创建一个套接字进行通信；Web是一种基于HTTP协议的互联网应用；一个域名可以对应多个IP地址，一个IP地址也可以对应多个域名

|  |
| --- |
| 得分 |
|  |

## 第八题（10分）并发

有三个线程PA、PB、PC协作工作以解决文件打印问题：PA将记录从磁盘读入内存缓冲区Buff1，每执行一次读一个记录；PB将缓冲区Buff1的内容复制到缓冲区Buff2，每执行一次复制一个记录； PC将缓冲区Buff2的内容打印出来，每执行一次打印一个记录。缓冲区Buff1可以放4个记录；缓冲区Buff2可以放8个记录。请用信号量及P、V操作实现上述三个线程以保证文件的正确打印。

PA() {

while(1) {

①

从磁盘读入一个记录

②

将记录放入Buff1

③

}

}

PB() {

while(1) {

④

从Buff1中取出一个记录

⑤

将记录放入Buff2

⑥

}

}

PC() {

while(1) {

⑦

从Buff2中取出一个记录

⑧

打印

}

}

1．请设计若干信号量，给出每一个信号量的作用和初值。

2．请将信号量上对应的PV操作填写在代码中适当位置。注意：每一标号处可以不填入语句（请标记成 X），或填入一条或多条语句。

|  |  |
| --- | --- |
| **标号** | **对应的操作** |
|  |  |
| ② |  |
| ③ |  |
| ④ |  |
| ⑤ |  |
| ⑥ |  |
| ⑦ |  |
| ⑧ |  |

答案：（本大题总得分按四舍五入取整数）

1．3分，答对1个0.5分

empty1，初值4；

full1，初值0；

empty2，初值8；

full2，初值0；

mutex1，初值1；

mutex2，初值2

2．7分

|  |  |
| --- | --- |
| **标号** | **对应的操作** |
|  | X |
| ② | P(&empty1);  P(&mutex1);  注意：顺序不能错，各0.5分 |
| ③ | V(&full1);  V(&mutex1);  各0.5分 |
| ④ | P(&full1);  P(&mutex1);  注意：顺序不能错，各0.5分 |
| ⑤ | P(&empty2);  P(&mutex2);  注意：顺序不能错，各0.5分 |
| ⑥ | V(&mutex2)  V(&full2)  各0.5分 |
| ⑦ | P(&full2);  P(&mutex2);  注意：顺序不能错，各0.5分 |
| ⑧ | V(&mutex2)  V(&empty2)  各0.5分 |