源代码：

1.1Getpid1.c:

#include<stdio.h>

#include<unistd.h>

Int main ()

{

Pid\_t pid();

Pid = getpid();

Printf(“%d\n”,pid);

Return 0;

}

Getpid2.c:

#include <stdio.h>

#include<unistd.h>

Int main()

{

Pid\_t pid;

Asm volatile(

“mov $0,%%ebx\n\t”

“int $0x80\n\t”

“mov %%eax,%0\n\t”

:”=”(pid)

);

Printf(“%d\n”,pid);

Return 0;

}

1.2Linux系统调用的C函数形式：

1. #include "unistd.h"
2. **int** main(){
3. write(1,"Hello World!",12);
4. **return** 0;
5. }

汇编代码：

1. section .data                           ;数据声明
2. msg     db      "Hello, world!",0xA     ;要输出的字符串
3. len     equ     $ - msg                 ;字符串长度
4. section .text                           ;代码段声明
5. global \_start       ;指定入口函数
7. \_start:                 ;函数在屏幕上输出Hello World！
8. mov     eax,4    ;系统调用号 (sys\_write)
9. mov     ebx,1    ;第一个参数，输出文件描述符
10. mov     ecx,msg    ;第二个参数，Hello World！字符串
11. mov     edx,len      ;第三个参数，字符串长度
12. **int**     0x80     ;调用内核功能
14. mov     eax,1     ;系统调用 (sys\_exit)
15. xor     ebx,ebx     ;第一个参数，退出返回码

**int**     0x80     ;调用内核功能

2.Cpu.c:

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include <assert.h>

int main(int argc, char \*argv[])

{

if (argc != 2) {

fprintf(stderr, "usage: cpu <string>\n");

exit(1);

}

char \*str = argv[1];

while (1) {

spin(1);

printf("%s\n", str);

}

return 0;}

3.mem.c:

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char \*argv[])

{

int \*p = malloc(sizeof(int)); // a1

assert(p != NULL);

printf("(%d) address pointed to by p: %p\n",

getpid(), p); // a2

\*p = 0; // a3

while (1) {

\*p = \*p + 1;

printf("(%d) p: %d\n", getpid(), \*p); // a4

}

return 0;

}

4.Thread.c:

#include <stdio.h>

#include <stdlib.h>

#include<thead.h>

volatile int counter = 0;

int loops;

void \*worker(void \*arg) {

int i;

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

counter++;

}

return NULL;

}

int main(int argc, char \*argv[])

{

if (argc != 2) {

fprintf(stderr, "usage: threads <value>\n");

exit(1);

}

loops = atoi(argv[1]);

pthread\_t p1, p2;

printf("Initial value : %d\n", counter);

pthread\_create(&p1, NULL, worker, NULL);

pthread\_create(&p2, NULL, worker, NULL);

pthread\_join(p1, NULL);

Pthread\_join(p2, NULL);

printf("Final value : %d\n", counter);

return 0;

}