Bernstein条件

## 操作系统实验(一)

---

### 1. 实验目的

通过进程的创建、撤消和运行加深对进程概念和进程并发执行的理解，明确进程与程序之间的区别。

### 2. 实验内容

(1) 了解系统调用[fork()](https://www.cnblogs.com/dongguolei/p/8086346.html)、exec()、exit()和waitpid()的功能和实现过程。

(2) 编写一段程序，使用系统调用fork()来创建两个子进程，并由父进程重复显示字符

串“parent:”和自己的进程号，子进程则重复显示字符串"child： ”和自己的进程号。

```

//t1\_1.c

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main(){

int p1,p2 ;

p1 = fork();

while((p1 = fork()) == -1);

if(p1 == 0)

{

printf("child1:%d\n",getpid());

}

else{

p2 = fork();

while((p2 = fork()) == -1);

if(p2 == 0)

{

printf("child2:%d\n",getpid());

}

else

{

printf("parent:%d\n",getpid());

}

}

return -1;

}

```

(3) 编写一段程序，使用系统调用fork()来创建一个子进程。子进程通过系统调用exec()

更换自己的执行代码，新的代码显示“ new program.”后，调用exit()结束。父进程则调用

Waitpid()等待子进程结束，并在子进程结束后，显示子进程的标识符然后正常结束。

```

// t1\_2.c

#include<stdio.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

#include<stdlib.h>

int main()

{

int p;

while((p = fork()) == -1);

if(p == 0)

{

printf("child's pid: %d\n",getpid());

execlp("./newpro","",NULL,NULL);

}

else{

printf("parent = %d\n",getpid());

int chpid = waitpid(p,NULL,0);

printf("child = %d\n",chpid);

exit(0);

}

}

// newpro.c

#include<stdio.h>

#include<stdlib.h>

int main()

{

printf("new program.\n");

exit(0);

}

```

(4)设计多进程，通过实验得出结果：父子进程能不能共享局部变量，共享全局变量？（即在一个进程中是否能修改另一个进程的数据），并用学过的相关进程知识解释.

```

// t1\_3.c

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h>

int global = 1;

int main()

{

int local = 0;

pid\_t p;

while((p = fork()) == -1);

if(p == 0)

{

global++;

local++;

printf("child global:%d,local:%d\n",global,local);

}

else{

waitpid(p,NULL,0);

printf("parent global:%d,local:%d\n",global,local);

}

return 0;

}

```