**实验1.1**

**步骤一：**

#include<stdio.h>

#include <pthread.h>

#include <unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<stdlib.h>

int main()

{

pid\_t pid,pid1;

pid=fork();

if(pid<0){

fprintf(stderr,"Fork Failed");

return 1;

}

else if(pid==0){

pid1=getpid();

printf("child: pid =%d",pid);

printf("child: pid1 =%d",pid1);

}

else {

pid1=getpid();

printf("parent: pid =%d",pid);

printf("parent: pid1 =%d",pid1);

wait(NULL);

}

return 0;

}

**步骤二：**

#include<stdio.h>

#include <pthread.h>

#include <unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<stdlib.h>

int main()

{

pid\_t pid,pid1;

pid=fork();

if(pid<0){

fprintf(stderr,"Fork Failed");

return 1;

}

else if(pid==0){

pid1=getpid();

printf("child: pid =%d",pid);

printf("child: pid1 =%d",pid1);

}

else {

pid1=getpid();

printf("parent: pid =%d",pid);

printf("parent: pid1 =%d",pid1);

}

return 0;

}

**步骤三：**

#include<stdio.h>

#include <pthread.h>

#include <unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<stdlib.h>

int value=0;

int main()

{

pid\_t pid,pid1;

pid=fork();

if(pid<0){

fprintf(stderr,"Fork Failed");

return 1;

}

else if(pid==0){

pid1=getpid(); value++;

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

printf("child: \*value =%p\n",&value);

}

else {

pid1=getpid(); value--;

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

printf("parent: \*value =%p\n",&value);

}

return 0;

}

**步骤四：**

#include<stdio.h>

#include <unistd.h>

#include<stdlib.h>

int value=0;

int main(){

pid\_t pid,pid1;

pid=fork();

if(pid<0){

fprintf(stderr,"Fork Failed");

return 1;

}

else if(pid==0){

pid1=getpid(); value++;

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

printf("child: \*value =%p\n",&value);

}

else {

pid1=getpid(); value--;

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

printf("parent: \*value =%p\n",&value);

}

value=value+5;

printf(" before return value=%d,\*value=%p\n",value,&value);

return 0;

}

**步骤五：**

子进程调用system函数

#include<stdio.h>

#include <pthread.h>

#include <unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<stdlib.h>

int main()

{

pid\_t pid,pid1;

pid=fork();

if(pid<0){

fprintf(stderr,"Fork Failed");

return 1;

}

else if(pid==0){

pid1=getpid();

printf("child process1 PID:%d\n",pid1);

system("/usr/local/src/system\_call");

printf("child process PID:%d\n",pid1);

}

else {

pid1=getpid();

printf("parent process PID:%d\n",pid1);

}

return 0;

}

子进程调用exce函数

#include<stdio.h>

#include <pthread.h>

#include <unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<stdlib.h>

int main()

{

pid\_t pid,pid1;

pid=fork();

if(pid<0){

fprintf(stderr,"Fork Failed");

return 1;

}

else if(pid==0){

pid1=getpid();

printf("child process1 PID:%d\n",pid1);

execl("/bin/sh", "sh", "-c", "./system\_call", (char \*)0);

printf("child process PID:%d\n",pid1);

}

else {

pid1=getpid();

printf("parent process PID:%d\n",pid1);

wait(NULL);

}

return 0;

}

system\_call.c源文件：

#include<stdio.h>

#include <pthread.h>

#include <unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<stdlib.h>

int main()

{

int pid=getpid();

printf("system\_call PID:%d\n",pid);

return 0;

}