1. 通过fork创建子进程，让父子进程都打印pid，都处于while(1),然后用ps进程查看，查看完毕，用kill命令杀掉

#include <func.h>

int main()

{

    pid\_t pid;

    pid=fork();

    if(0==pid)

    {

        //孩子的代码写这里

        printf("I am child,pid=%d,ppid=%d\n",getpid(),getppid());

        while(1);

        return 0;

    }

    else {

        //父亲的代码写这里

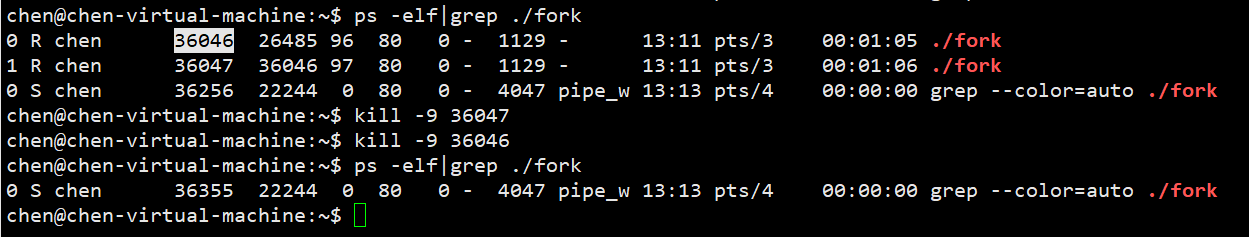
        printf("I am parent,mychild=%d,pid=%d,ppif=%d",pid,getpid(),getppid());

        while(1);

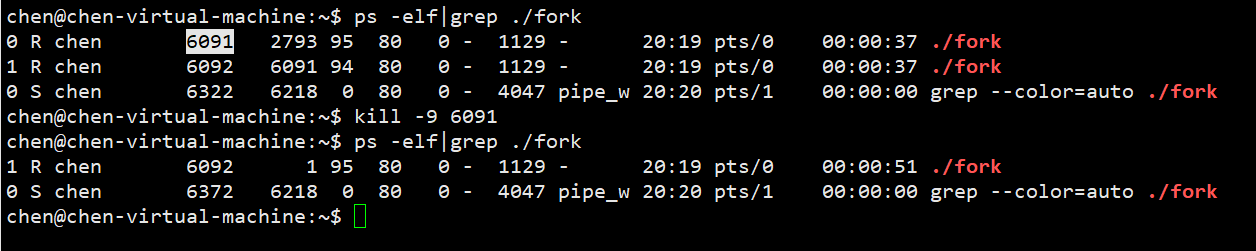
        return 0;

    }

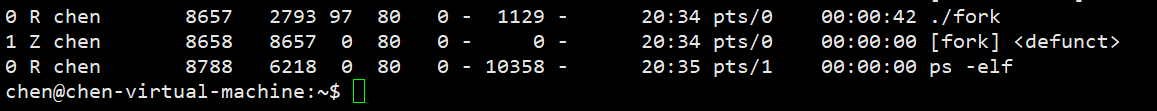
}



1. 能够得到孤儿进程；



1. 通过fork创建进程，能够产生僵尸进程；



1. 通过fork创建子进程，然后父进程通过wait回收，同时拿到子进程的返回值

#include <func.h>

int main()

{

    pid\_t pid;

    pid=fork();

    if(0==pid)

    {

        //孩子的代码写这里

        printf("I am child,pid=%d,ppid=%d\n",getpid(),getppid());

        exit(3);

    }

    else {

        //父亲的代码写这里

        printf("I am parent,mychild=%d,pid=%d,ppif=%d",pid,getpid(),getppid());

        int status;

        pid=wait(&status);

        if(WIFEXITED(status))

        {

            printf("child exit code=%d\n",WEXITSTATUS(status));

            printf("%d\n",status);

            printf("%d\n",WIFEXITED(status));

        }

        else{

            printf("child crash\n");

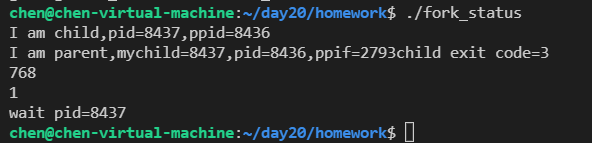
        }

        printf("wait pid=%d\n",pid);

        return 0;

    }

}



5、通过execl拉起来一个add进程，并给其传参3和4，让其求和最终得到7

#include <func.h>

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

{

    ARGS\_CHECK(argc,3)

    int a=atoi(argv[1]);

    int b=atoi(argv[2]);

    printf("%d+%d=%d\n",a,b,a+b);

    return 0;

}

#include <func.h>

int main()

{

    execl("./add","add","3","4",NULL);

    return 0;

}

