系统调用fork创建进程，复制父进程的数据、堆栈、进程环境。 但是新开辟地址。

vfork共享。

int fork 返回0 <0 >0 if(fork()) {父进程执行程序}else{子进程执行程序} 共同（输出顺序不定）

父子执行同一程序，继承资源

线程 \_clone，共享数据段，独立堆栈

进程号PID(写入日志或创建临时文件)，父进程号PPID init 1 内核自举后第一个进程

wait() 父进程等待子进程撤销后唤醒否则睡眠

sleep() 进程睡眠等待n秒

exit() 终止进程唤醒wait的父进程

http://wenku.baidu.com/view/4f1f680903d8ce2f006623a4.html?from=search

相关函数：longjmp, siglongjmp, setjmp   
表头文件：#include <setjmp.h>   
函数定义：int sigsetjmp(sigjmp\_buf env, int savesigs)   
函数说明：sigsetjmp()会保存目前堆栈环境，然后将目前的地址作一个记号，而在程序其他地方调用siglongjmp()时便会直接跳到这个记号位置，然后还原堆栈，继续程序好执行。   
参数env为用来保存目前堆栈环境，一般声明为全局变量   
参数savesigs若为非0则代表搁置的信号集合也会一块保存   
当sigsetjmp()返回0时代表已经做好记号上，若返回非0则代表由siglongjmp（）跳转回来。   
返回值  ：返回0代表局促存好目前的堆栈环境，随时可供siglongjmp()调用， 若返回非0值则代表由siglongjmp()返回

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <sys/socket.h>

#include <resolv.h>

#include <stdlib.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#define MAXBUF 1024

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

{

int sockfd, len;

struct sockaddr\_in dest;

char buffer[MAXBUF + 1];

if (argc != 3)

{

printf(" error format,it must be:\n\t\t%s IP port\n",argv[0]);

exit(EXIT\_FAILURE);

}

if ((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

perror("Socket");

exit(errno);

}

printf("socket created\n");

bzero(&dest, sizeof(dest));

dest.sin\_family = AF\_INET;

dest.sin\_port = htons(atoi(argv[2]));

if (inet\_aton(argv[1], (struct in\_addr \*) &dest.sin\_addr.s\_addr) == 0)

{

perror(argv[1]); exit(errno);

}

if (connect(sockfd, (struct sockaddr \*) &dest, sizeof(dest))==-1)

{

perror("Connect ");

exit(errno);

}

printf("server connected\n");

pid\_t pid;

if(-1==(pid=fork()))

{

perror("fork");exit(EXIT\_FAILURE);

}

else if (pid==0)

{

while (1)

{

bzero(buffer, MAXBUF + 1);

len = recv(sockfd, buffer, MAXBUF, 0);

if (len > 0)

printf("recv successful:'%s',%d byte recv\n",buffer, len);

else if(len < 0)

{

perror("recv");

break;

}

else

{

printf("the other one close the connection\n");

break;

}

}

}

else

{

while (1)

{

bzero(buffer, MAXBUF + 1);

printf("message to send:");

fgets(buffer, MAXBUF, stdin);

if (!strncasecmp(buffer, "quit", 4))

{

printf(" i will quit!\n");

break;

}

len = send(sockfd, buffer, strlen(buffer) - 1, 0);

if (len < 0)

{

perror("send");

break;

}

}

}

close(sockfd);

return 0;

}