本地套接字编程（socket IPC，稳定性好）

### 1、基础知识

1）socket API原本是为网络通讯设计的，但后来在socket的框架上发展出一种IPC机制，就是UNIX Domain Socket。虽然网络socket也可用于同一台主机的进程间通讯（通过loopback地址127.0.0.1），但是UNIX Domain Socket用于IPC更有效率：不需要经过网络协议栈，不需要打包拆包、计算校验和、维护序号和应答等，只是将应用层数据从一个进程拷贝到另一个进程。这是因为，IPC机制本质上是可靠的通讯，而网络协议是为不可靠的通讯设计的。面向消息的UNIX Domain Socket是可靠的，消息既不会丢失也不会顺序错乱。

2）UNIX Domain Socket是全双工的，API接口语义丰富，相比其它IPC机制有明显的优越性，目前已成为使用最广泛的IPC机制，比如X Window服务器和GUI程序之间就是通过UNIXDomain Socket通讯的。

### 2、编程要点

1）使用UNIX Domain Socket的过程和网络socket十分相似，也要先调用socket()创建一个socket文件描述符，address family指定为AF\_UNIX，type可以选择SOCK\_DGRAM或SOCK\_STREAM，protocol参数仍然指定为0即可。

2）UNIX Domain Socket与网络socket编程最明显的不同在于地址格式不同，用结构体sockaddr\_un表示，网络编程的socket地址是IP地址加端口号，而UNIX Domain Socket的地址是一个socket类型的文件在文件系统中的路径，这个socket文件由bind()调用创建，如果调用bind()时该文件已存在，则bind()错误返回。

struct sockaddr\_un {

\_\_kernel\_sa\_family\_t sun\_family; //地址结构类型

char sun\_path[UNIX\_PATH\_MAX]; //socket文件名(含路径)

};

3）在需要传入sockaddr\_un结构体长度的函数（bind()和connect()）中，不应该使用sizeof(struct sockaddr\_un)，而应该求出“地址结构类型长度+ socket文件名的真实长度”，具体求法如下：

len = offsetof(struct sockaddr\_un, sun\_path) + strlen(name);

注：#define offsetof(type, member) ((int)&((type \*)0)->MEMBER)

offsetof(struct sockaddr\_un, sun\_path) 求出在结构体中sun\_path变量的偏移位置，即sun\_family变量占用空间大小。

### 3、具体实现

1）server：

**#include <stdio.h>**

**#include <unistd.h>**

**#include <sys/socket.h>**

**#include <strings.h>**

**#include <string.h>**

**#include <ctype.h>**

**#include <arpa/inet.h>**

**#include <sys/un.h>**

**#include <stddef.h>**

**#define SERV\_ADDR "serv.socket"**

**int main(void)**

**{**

**int lfd, cfd, len, size, i;**

**struct sockaddr\_un servaddr, cliaddr;**

**char buf[4096];**

**lfd = socket(AF\_UNIX, SOCK\_STREAM, 0);**

**bzero(&servaddr, sizeof(servaddr));**

**servaddr.sun\_family = AF\_UNIX;**

**strcpy(servaddr.sun\_path,SERV\_ADDR);**

**len = offsetof(struct sockaddr\_un, sun\_path) + strlen(servaddr.sun\_path);**

**unlink(SERV\_ADDR);**

**bind(lfd, (struct sockaddr \*)&servaddr, len); /\* 参3不能是sizeof(servaddr) \*/**

**listen(lfd, 20);**

**printf("Accept ...\n");**

**while (1) {**

**len = sizeof(cliaddr);**

**cfd = accept(lfd, (struct sockaddr \*)&cliaddr, (socklen\_t \*)&len);**

**len -= offsetof(struct sockaddr\_un, sun\_path); /\* 得到文件名的长度 \*/**

**cliaddr.sun\_path[len] = '\0'; /\* 确保打印时,没有乱码出现 \*/**

**printf("client bind filename %s\n", cliaddr.sun\_path);**

**while ((size = read(cfd, buf, sizeof(buf))) > 0) {**

**for (i = 0; i < size; i++)**

**buf[i] = toupper(buf[i]);**

**write(cfd, buf, size);**

**}**

**close(cfd);**

**}**

**close(lfd);**

**return 0;**

**}**

2）client：

**#include <stdio.h>**

**#include <unistd.h>**

**#include <sys/types.h>**

**#include <sys/socket.h>**

**#include <strings.h>**

**#include <string.h>**

**#include <ctype.h>**

**#include <arpa/inet.h>**

**#include <sys/un.h>**

**#include <stddef.h>**

**#define SERV\_ADDR "serv.socket"**

**#define CLIE\_ADDR "clie.socket"**

**int main(void)**

**{**

**int cfd, len;**

**struct sockaddr\_un servaddr, cliaddr;**

**char buf[4096];**

**cfd = socket(AF\_UNIX, SOCK\_STREAM, 0);**

**bzero(&cliaddr, sizeof(cliaddr));**

**cliaddr.sun\_family = AF\_UNIX;**

**strcpy(cliaddr.sun\_path,CLIE\_ADDR);**

**len = offsetof(struct sockaddr\_un, sun\_path) + strlen(cliaddr.sun\_path);**

**unlink(CLIE\_ADDR);**

**bind(cfd, (struct sockaddr \*)&cliaddr, len); /\* 客户端也需要bind \*/**

**bzero(&servaddr, sizeof(servaddr)); /\* 构造server 地址 \*/**

**servaddr.sun\_family = AF\_UNIX;**

**strcpy(servaddr.sun\_path,SERV\_ADDR);**

**len = offsetof(struct sockaddr\_un, sun\_path) + strlen(servaddr.sun\_path);**

**connect(cfd, (struct sockaddr \*)&servaddr, len);**

**while (fgets(buf, sizeof(buf), stdin) != NULL) {**

**write(cfd, buf, strlen(buf));**

**len = read(cfd, buf, sizeof(buf));**

**write(STDOUT\_FILENO, buf, len);**

**}**

**close(cfd);**

**return 0;**

**}**