```
/*
 * isfilter.c
 *
 * iSocketFilter (iSF) - implantable socket filter
 * (c) 2020 ITS more Co., Ltd., // Licenced by CC(BY)
 * v1.0 - 2020-0805 created by SatoxITS (sato@its.more.jp)
 */
#define _ISFVER   "iSocketFilter/1.0 (2020-0805 (^-^)//" __DATE__ __TIME__ ")"
#define _ISFMARK  "##iSF(^-^)/"  // a mark to distinguish this, can be "" (^-^)
#define _ISFENV   "_ISFENV"      // environment variable to control this code
#define _ISFBFSZ  (256*1024)     // this size seems enough
#include <string.h>

#define _ISFFLAG(conf) (getenv(_ISFENV) && strstr(getenv(_ISFENV),conf) != 0)

#if defined(SSH_SERVICE_NAME) // maybe used in SSH
// Usage: (1) include this code in sshconnect.c:ssh_connect_direct()
//  just before calling ssh_packet_set_connection().
//  (2) invoke ssh with _ISFENV environment variable defined
#define _ISFDBG(fmt,...) debug(_ISFMARK fmt,##__VA_ARGS__)
#define _ISFDBGV(fmt,...) (_ISFFLAG("V")?_ISFDBG(fmt,##__VA_ARGS__):0)
#endif

#if _ISFFUNC
int _ISFFUNC(sock){
#endif
  if( _ISFFLAG("") ){
        char buf[_ISFBFSZ];
        int pid = getpid();
        int Exsock = sock; // socket to an external connection (to be given)
        int Insocks[2]; // External[Exsock] - RelayS - [0]Insock[1] - Internal
        int Insock;
        int serno = 0;
        int rcc;
        int wcc;
        //int xf1,xf2,xf3;
        //xf1 = dup(0); xf2 = dup(0); xf3 = dup(0);

        _ISFDBG("[%d] %s",pid,_ISFVER);
        Insocks[0] = Insocks[1] = -1;
        socketpair(AF_UNIX, SOCK_STREAM, 0, Insocks);
        Insock = Insocks[0];

        pid = fork();
        if( pid == 0 ){
                close(Insocks[1]);
                pid = getpid();
```

```
				_ISFDBG("[%d] relay to Ex[%d] <- Lo[%d]",pid,Exsock,Insock);
				for(;;){
					errno = 0;
					rcc = read(Insock,buf,sizeof(buf));
					_ISFDBGV("[%d] fromIn recv#%d = %d E%d(%s)",
						pid,++serno,rcc,errno,strerror(errno));
					if( rcc <= 0 ){
						break;
					}
					wcc = send(Exsock,buf,rcc,0);
					if( wcc != rcc ){
						break;
					}
				}
				_ISFDBG("[%d] relay to Ex end",pid);
				exit(0);
			}
			pid = fork();
			if( pid == 0 ){
				pid = getpid();
				close(Insocks[1]);
				_ISFDBG("[%d] relay from Ex[%d] -> Lo[%d]",pid,Exsock,Insock);
				for(;;){
					errno = 0;
					rcc = recv(Exsock,buf,sizeof(buf),0);
					_ISFDBGV("[%d] fromEx recv#%d = %d E%d(%s)",
						pid,++serno,rcc,errno,strerror(errno));
					if( rcc <= 0 ){
						break;
					}
					wcc = send(Insock,buf,rcc,0);
					if( wcc != rcc ){
						break;
					}
				}
				_ISFDBG("[%d] relay from Ex end",pid);
				exit(0);
			}
			close(Insocks[0]);
			close(Exsock);
			sock = Insocks[1];
			//close(xf1); close(xf2); close(xf3);
  }
#if _ISFFUNC
  return sock;
}
void __exit(int status);
void exit(int status){
			wait(0);
```

```
        __exit(status);
}
#endif
```