1076

Bell Laboratories

Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication (see GEI 13.9-3)

Title- Lkout, a New UNIX System Call to Suspend and Resume Input from User Terminals

Date- December 17, 1975

TM- 75-4145-02

Other Keywords- UNIX

Author(s)	Location and Room	Extension
R. C. White, Jr.	HO 4C-614	6142

Charging Case- 39447-13

Filing Case- 40117

ABSTRACT

In order to develop a reasonable personnel subsystem for the file system dump and restore practices used in the RBCS project, a new UNIX system call to suspend and resume input from user terminals has been written. This system call allows the operating system to be placed in a quiet state without logging off users or terminating their current processes. When lkout is invoked in the suspend mode, succeeding input from user terminals is discarded. When lkout is invoked in the resume mode, succeeding input from user terminals proceeds in the normal way. System console input is never suspended. Input from DC11 ports is always discarded if the lkout suspend mode is in effect. If lkout is invoked from DH11 port, input from that port is never discarded. In addition, it is possible to exclude individual DH11 ports from the suspend process so that their input can continue uneffected. Level I commands using lkout have been written for use in the personnel subsystem shell procedures for dumping and restoring RBCS files.

Pages Text 5 Other 11 Total 17

No. Figures ____ No. Tables ____ No. Refs. 1

Address Label

Bell Laboratories

subject: Lkout, a New UNIX System Call to Suspend and Resume Input from User Terminals - Charging Case 39447-13, Filing Case 40117

date: December 17, 1975

from: R. C. White, Jr. TM-75-4145-02

MEMORANDUM FOR FILE

I. INTRODUCTION

The RBCS file system in the Miami trial is dumped twice daily and must be done when no other processes are active. Since all users are connected via private line data sets it is not possible to easily busy out the modems to keep users from logging in nor is it acceptable from a personnel subsystem viewpoint to call all users and ask them to log-off. Since many users use protected forms, it is unacceptable to send a message to their CRT asking them to log-off. Also it was decided not to arbitrarily kill user processes and log them off the system. As a result of these constraints a new UNIX system call, lkout, was written which can suspend and resume input from user terminals without logging them off or killing their current processes. This mode of operation is particularly attractive to the RBCS project where there are many personnel practices which make heavy use of shell procedures. If such shell procedures are prematurely terminated, data base corruption can easily result. The lkout system call allows the shell procedures to to be placed in a temporarily suspended state from which they can continue without ill effects.

II. FEATURES OF THE LKOUT SYSTEM CALL

The lkout system call is used in conjunction with modified DC11 and DH11 drivers. These drivers have had their receive character interrupt routines modified to test the value of the global integer "rbcslock." In the case of the DC11 driver if rbcslock is nonzero all received characters are simply discarded. In the case of the DH11 driver it is possible to exclude certain ports from the effects of lkout. The DH11 driver looks at individual bits in rbcslock and if the bit corresponding to the port number of the received character is clear the character is accepted otherwise the character is discarded. As an example, if bit 03 in rbcslock is set, then characters received on DH11 line 03

will be accepted as usual. Lkout never discards characters at the DH11 port from which the lkout call is invoked. Lkout also works on Dataspeed 40 ports by withholding supervisory transmitted data if the DH11 line is to be suspended. The checking of rbcslock for Dataspeed 40 ports is handled in the modem interrupt routine. In the case of the Dataspeed 40 a record is kept in the global integer "rbcswstd" of terminals for which supervisory transmitted data has been withheld so that when input is allowed to resume, the supervisory transmit signal can be sent to all waiting Dataspeed 40 terminals.

The system console which uses the KL11 driver is never effected by the lkout system call.

Obviously lkout must be carefully used or the work of users can be fatally restricted. The RBCS project uses the UNIX cron program to check once every 15 minutes to see if lkout is active and if it is, a warning message is printed on the system console.

The lkout system call is supported by two level I commands for use in the RBCS dump and restore shell procedures. These commands are suspend for activating lkout and resume for returning to normal operations. Suspend is used as follows:

$$\text{suspend } [n_i] \qquad 0 \leq n_i \leq 15$$

where n_i is the minor device number of the DH11 port or ports which will not have their input suspended. Resume is trivial to use, the command is simply: resume.

III. INSTALLING LKOUT IN UNIX

A. The level II lkout.c system call is compiled and archived in /usr/sys/lib2. The source code for lkout.c is in Appendix I.

B. The assembly language interface to the C compiler lkoutc.s is assembled and archived in /lib/libc.a. The source code for lkoutc.s is in Appendix II.

C. The level I command suspend.c is compiled and the a.out file put in some out-of-the-way place such as /etc/suspend. The source code is in Appendix III.

D. Similarly the level I command resume.c is compiled and put in /etc/resume. The source code is in Appendix IV.

E. The system entry table sysent.c must be edited to recognize the lkout system call (lkout = 51), recompiled, and archived in /usr/sys/libl. The code for sysent.c is shown in Appendix V.

F. Recognition of the major and minor port device numbers required for the selective suspension features required the addition of a new member to the tty structure. This is indicated by the arrow in the new tty.h shown in Appendix VI. All UNIX code that references /usr/sys/tty.h must be recompiled.

G. The code changes to dc.c and dh40.c were relatively short so only the code of the particular routines effected in each driver is reproduced. The dh40.c sections are in Appendix VII, Appendix VIII contains the dc.c changes. Both dh40.c and dc.c must be recompiled and archived in /usr/sys/lib2.

H. A new UNIX must be loaded and the system rebooted.

R. C. White, Jr.

HO-4145-RCW-jm

Atts.
References
Appendices I through VIII

REFERENCES

1. R. C. White, Jr. "UNIX DH-11 Driver to Support Both Teletypewriter and Dataspeed 40 Timesharing Terminals, TM-75-4145-01, December 17, 1975.

```
#
/*
 *      Copyright 1975 Bell Telephone Laboratories Inc.
 */

/*
 *      UNIX System Call to Suspend and Resume Reception of Data from
 *      Remote Terminals
 */

#include "/usr/sys/param.h"
#include "/usr/sys/reg.h"
#include "/usr/sys/user.h"
#include "/usr/sys/tty.h"
#include "/usr/sys/proc.h"
#include "/usr/sys/conf.h"

#define SUSPEND 0
#define RESUME 1
#define NDH11 16
#define SUPTD 010
#define DH11 4

lkout()
{
        register struct tty *tp;
        register int i,mask;
        int mode,ports;
        extern int rbcslock,rbcswstd;
        extern struct tty dh11[NDH11];

        tp= u.u_procp->p_ttyp;
        mode= u.u_arg[0];
        ports= u.u_arg[1];
        if(mode == SUSPEND){
                if(tp->t_dev.d_major == DH11) ports=& ~(01<<(tp->t_dev.d_minor));
                rbcslock= ports;
                u.u_ar0[R0]= 0;
                return;
        }
        if(mode == RESUME){
                rbcslock= 0;
                mask= 01;
                for( i=0 ; i<NDH11 ; i++ ){
                        if(mask & rbcswstd){
                                rbcswstd=& ~mask;
                                dmstat(&dh11[i],'s',SUPTD,0);
                        }
                        mask= mask<<1;
                }
                u.u_ar0[R0]= 0;
                return;
        }
        u.u_ar0[R0]= -1;        /* error */
        return;
}
```

```
/ C Library -- lkout

/ value = lkout(mode,ports)

.globl _lkout,retrn,cerror

        lkout= 51.
_lkout:
        mov r5,-(sp)
        mov sp,r5
        mov 4(r5),0f
        mov 6(r5),0f+2
        sys 0; 9f
        bec 1f
        jmp cerror
1:
        jmp retrn
.data
9:
        sys lkout; 0:..;..
```

APPENDIX III

```
#
/*
 * Suspend users preparatory to dumping
 */

#define SUSPEND 0
#define ALL 0177777

main(argc,argv)
int argc;
char *argv[];
{
        int ports,tmp;

        printf("Locking out user input\n");
        ports= ALL;
        while(--argc){
                tmp= number(*++argv);
                if((tmp < 0) || (tmp > 15)) continue;
                        else ports=& ~(01<<tmp);
                printf("Port %d not locked out.\n",tmp);
        }
        exit(lkout(SUSPEND,ports));
}
```

```
#
/*
 * resume users after to dumping
 */

#define RESUME 1

main() {
    printf("Resuming user input\n");
    exit(1kout(RESUME,0));
}
```

APPENDIX V

```
#
/*
 *      Copyright 1973 Bell Telephone Laboratories Inc
 */

int     svsent()
{
        0, &nullsys,                    /*  0 = indir */
        0, &rexit,                      /*  1 = exit */
        0, &fork,                       /*  2 = fork */
        2, &read,                       /*  3 = read */
        2, &write,                      /*  4 = write */
        2, &open,                       /*  5 = open */
        0, &close,                      /*  6 = close */
        0, &wait,                       /*  7 = wait */
        2, &creat,                      /*  8 = creat */
        2, &link,                       /*  9 = link */
        1, &unlink,                     /* 10 = unlink */
        2, &exec,                       /* 11 = exec */
        1, &chdir,                      /* 12 = chdir */
        0, &gtime,                      /* 13 = time */
        3, &mknod,                      /* 14 = mknod */
        2, &chmod,                      /* 15 = chmod */
        2, &chown,                      /* 16 = chown */
        1, &sbreak,                     /* 17 = break */
        2, &stat,                       /* 18 = stat */
        2, &seek,                       /* 19 = seek */
        0, &getpid,                     /* 20 = getpid */
        3, &smount,                     /* 21 = mount */
        1, &sumount,                    /* 22 = umount */
        0, &setuid,                     /* 23 = setuid */
        0, &getuid,                     /* 24 = getuid */
        0, &stime,                      /* 25 = stime */
        0, &nosys,                      /* 26 = x */
        0, &nosys,                      /* 27 = x */
        1, &fstat,                      /* 28 = fstat */
        0, &nosys,                      /* 29 = x */
        1, &smdate,                     /* 30 = smdate */
        1, &stty,                       /* 31 = stty */
        1, &gtty,                       /* 32 = gtty */
        0, &nosys,                      /* 33 = x */
        0, &nice,                       /* 34 = nice */
        0, &ssleep,                     /* 35 = sleep */
        0, &sync,                       /* 36 = sync */
        1, &kill,                       /* 37 = kill */
        0, &getswit,                    /* 38 = switch */
        0, &nosys,                      /* 39 = x */
        0, &nosys,                      /* 40 = x */
        0, &dup,                        /* 41 = dup */
        0, &pipe,                       /* 42 = pipe */
        1, &times,                      /* 43 = times */
        4, &profil,                     /* 44 = prof */
        0, &nosys,                      /* 45 = tiu */
        0, &setgid,                     /* 46 = setgid */
        0, &getgid,                     /* 47 = getgid */
        2, &ssig,                       /* 48 = sig */
```

```
        2, &sem,                /* 49 = sem */
        3, &kall,               /* 50 = kall */
        2, &lkout,              /* 51 = lkout */
        0, &nosys,              /* 52 = x */
        0, &nosys,              /* 53 = x */
        0, &nosys,              /* 54 = x */
        0, &nosys,              /* 55 = x */
        0, &nosys,              /* 56 = x */
        0, &nosys,              /* 57 = x */
        0, &nosys,              /* 58 = x */
        0, &nosys,              /* 59 = x */
        0, &nosys,              /* 60 = x */
        0, &nosys,              /* 61 = x */
        0, &nosys,              /* 62 = x */
        0, &nosys               /* 63 = x */
};
```

```
struct clist {
        int     c_cc;
        int     c_cf;
        int     c_cl;
};

struct tty {
        struct  clist t_rawq;
        struct  clist t_canq;
        struct  clist t_outq;
        int     t_flags;
        int     *t_addr;
        char    t_delct;
        char    t_col;
        char    t_intrup;
        char    t_quit;
        int     t_state;
        char    t_char;
        int     t_speeds;
        int     t_dhaddr;
        char    t_stopct;
        char    t_offct;
        int     t_dev;
};

char partab[];

#define TTIPRI  10
#define TTOPRI  20
#define CERASE  '#'
#define CEOT    004
#define CETX    003
#define CDC2    022
#define CDC4    024
#define CTAB    011
#define CKILL   '@'

/* limits */
#define TTHIWAT 50
#define TTLOWAT 30
#define TTYHOG  256

/* modes */
#define NODELAY 01
#define XTABS   02
#define LCASE   04
#define ECHO    010
#define CRMOD   020
#define RAW     040
#define ODDP    0100
#define EVENP   0200
#define T2741   0400
#define OCURR   02000
#define ICURR   04000
#define MTOPL   010000
#define TBDELAY 0400
```

```
#define BLOCK   01000

/* Bits */
#define DONE    0200
#define IENABLE 0100
#define TIMEOUT 01
#define WOPEN   02
#define ISOPEN  04
#define SSTART  010
#define CARR_ON 020
#define BUSY    040
#define REC_ON  0100
#define TRAN_ON 0200
#define SRDBIT  0400
#define STDBIT  01000
#define SUPTD 010
#define ENDBIT  02000
#define DELAY40     04000
#define NAK377  010000
#define XR_ON   020000
#define SRD_ON  040000
#define STD_ON  0100000
#define DELTA   1
#define PAUSE   18
#define ONE     60
#define TWO     120
#define THREE   180
#define TEN     600
#define FIFTEEN 900
```

APPENDIX VII

```
int rocslock,rocswstd;

dhrint()
{
        register struct tty *tp;
        register int c;

        while ((c = DHADDR->dhnxch) < 0) {      /* char. present */
                tp = &dh11[(c>>8)&017];
                if (tp >= &dh11[NDH11])
                        continue;
                if(((tp->t_state & ISOPEN) == 0) ||
                        (c&PERROR) ||
                        ((tp->t_state & CARR_ON) != 0)){
                                wakeup(&tp->t_state);
                                continue;
                }
                if (c&FRERROR)          /* break. return null */
                        c = 0;
                if(((tp->t_flags & MOD40) == 0) &&
                        (rocslock & (01<<((c>>8)&017)))) continue;
                        else ttyinput(c,tp);
                if((tp->t_rawq.c_cc >= 90) &&
                        ((tp->t_flags & MOD40) != 0) &&
                        ((tp->t_state & STOBIT) == 0)){
                                dmstat(tp,'c',SUPTD,0);
                                tp->t_state=| STOBIT;
                                if((tp->t_state & WAK377) == 0){
                                        if(putc(0377,&tp->t_rawq) == 0){
                                                tp->t_delct++;
                                                tp->t_state=| WAK377;
                                        }
                                }
                                wakeup(&tp->t_rawq);
                }
        }
}

dmint()
{
        register struct tty *tp;
        register csr,lsr;
        int scratch;
        extern hup();

        if((DMADDR->dmcsr & DONE) != 0){
                tp= &dh11[DMADDR->dmcsr&017];
                if(tp < &dh11[ndh11]){
                        csr= DMADDR->dmcsr;
                        lsr= DMADDR->dmlstat;
                        if((tp->t_flags & MOD40) == 0)
                                if ((DMADDR->dmlstat&CARRIER)==0) {
                                        tp->t_state =& ~CARR_ON;
                                        if((tp->t_state & WOPEN) == 0){
                                                tp->t_state=& ~ISOPEN;
                                                signal(tp, SIGHUP);
```

```
                                        DMADDR->dmlstat =& ~CDLEAD;
                                        flushttv(tp);
                                }
                        } else {
                                tp->t_state=| CARR_ON;
                                wakeup(&tp->t_state);
                        }
                else{
                        if((csr & CTRANS) != 0){
                                if((lsr & CARRIER) == 0){
                                        tp->t_state=& ~REC_ON;
                                        DMADDR->dmlstat=& ~SUPTD;
                                        if((tp->t_flags & BLOCK) != 0)
                                                if((tp->t_state & WAK377) == 0){
                                                        if(putc(0377,&tp->t_rawq) == 0)
                                                                tp->t_delct++;
                                                        if(putc(0377,&tp->t_rawq) == 0)
                                                                tp->t_delct++;
                                                } else if(putc(0377,&tp->t_rawq) == 0)
                                                                tp->t_delct++;
                                        wakeup(&tp->t_rawq);
                                        wakeup(&tp->t_state);
                                } else {
                                        tp->t_state=| REC_ON;
                                        scratch= 01<<(DMADDR->dmcsr&017);
                                        if(rbcslock & scratch) rbcswstd=| scratch;
                                                else DMADDR->dmlstat=| SUPTD;
                                }
                        }
                        if((csr & CSTRANS) != 0){
                                if((lsr & CLSEND) != 0) tp->t_state=| TRAN_ON;
                                wakeup(&tp->t_state);
                        }
                }
                if((csr & SRTRANS) != 0)
                        if((lsr & SUPRD) == 0){
                                tp->t_state=& ~SRD_ON;
                                if((tp->t_state & (REC_ON|TRAN_ON|WOPEN)) == 0)
                                        timeout(hup,tp,TEN);
                        } else {
                                tp->t_state=| SRD_ON|CARR_ON;
                                wakeup(&tp->t_state);
                        }
        }
        dmc= DMADDR->dmcsr;
        dms= DMADDR->dmlstat;
        DMADDP->dmcsr= IENABLE|SCENABL;
    }
}
```

```
dcrint(dev)
{
        register struct tty *tp;
        register int c, csr;
        extern int rbcslock;

        tp = &dc11[dev.d_minor];
        c = tp->t_addr->dcrbuf;
        if (((csr = tp->t_addr->dcrcsr) & CARRIER) == 0) {
                if ((tp->t_state&WOPEN) == 0) {
                        tp->t_addr->dcrcsr =& ~CDLEAD;
                        if (tp->t_state & CARR_ON)
                                signal(tp, SIGHUP);
                        flushtty(tp);
                }
                tp->t_state =& ~CARR_ON;
                return;
        }
        if (csr&ERROR || (tp->t_state&ISOPEN)==0) {
                if (tp->t_state&WOPEN && csr&CARRIER)
                        tp->t_state =| CARR_ON;
                wakeup(tp);
                return;
        }
        csr =& PARITY;
        if (csr&&(tp->t_flags&ODDP) || !csr&&(tp->t_flags&EVENP))
                if(!rbcslock) ttyinput(c,tp);
}
```