;login:

THE UNIX NEWSLETTER

LUME 3 NUMBER 1 DECEMBER-JANUARY 1978

HERE WE ARE

Here at long last, with many apologies, is the "first" issue of ;login: for 1978. The "March" issue will go to the printer within a week or so and then "May" will complete the catching-up on old correspondence.

INVOICE

Enclosed is an invoice for eighteen months for January 1978 - 1979. As was explained in the recently mailed report of the ?NIX committee", the annual dues starting July 1978 is $50.00. We feel honor bound to charge $10.00 per year for the period before July 1, hence the invoice is for $5.00 for the first six months and for $25.00 for each of the next six month periods. By striking out the appropriate lines you may pay for 6, 12, or 18 months. If you can avoid processing purchase orders we will be most grateful.

ADDRESS CHANGE

Effective with receipt of this issue, all correspondence should be adressed as indicated in the box below. We are trying to "get our act together".

Address editorial material, payments, and software submission to
Melvin Ferentz
Box 8
The Rockefeller University
1230 York Avenue
New York, N.Y. 10021

Subscription requests and address changes should be addressed to
Armand Gazes
Box 8
The Rockefeller University
1230 York Avenue
New York, N.Y. 10021

*Trademark of Bell Laboratories

Institute of Animal Resource Ecology
The University of British Columbia
2075 Wesbrook Mall
Vancouver, B.C., Canada V6T 1W5

TEL. (604) 228-2731

December 7, 1977.

Prof. Melvin Ferentz,
c/o CUNY/UCC,
555 West 57th Street,
New York, N.Y. 10019

Dear Professor Ferentz:

After frustrating experiences with most of the available UNIX Fortran compilers (FC, DOS FORTRAN, RT-11 FTN) under UNIX, I finally started writing my own last July. Although not yet in a final documented form, I believe that it is now in good enough shape (i.e., better than FC by all measures) to distribute it on a pre-release basis to university UNIX installations.

The essential characteristics of the compiler are:

1. It implements the proposed Fortran 77 language standard
2. compatible with existing compilers for Fortran IV, except where superceeded by the Fortran 77 standard
3. written in C
4. implements optional length specifications (INTEGER *2, *4, REAL *4, *8, LOGICAL *1, *2, COMPLEX *8, *16)
5. -T switch switches integers from 2 to 4 by default length
6. very fast compile times
7. UNIX "a.out" format ".o" files are produced directly from the compiler (no "as" stage!)
8. UNIX AR format archieve is produced containing the objects if the source file contains more than one routine
9. requires about 28K words (its large!)
10. Good diagnostics and error messages (compiler and run time)
11. Subroutine calling sequence is DEC PC, and UNIX C compatible (at the same time!)
12. Threaded rather than executable code is produced.

My main purpsoes in making the compiler available are:

1. It will help me quickly locate remaining bugs

2. Others might be willing to work on those areas that are currently deficient in either the compiler or the run time support.
3. We are still stuck with Fortran, so we might as well have a reasonable compiler for it under UNIX.

Areas requiring additional work:

1. the COMPLEX arithmetic routines for the routine.
2. The I/O system. Currently a modified version of the UNIX PC routines are being used. At minimum the format processor has to be improved.
3. Not all of the standard fortran library routines are provided.
4. Some threaded routines are probably missing or buggy.
5. User level documentation is virtually non-existant (but then the same holds true for UNIX FC).

I expect to have a proper distribution of this compiler ready in about six months. Those who wish a pre-release copy should contact me at the above address or call me at 604 228-6527.

Bill Webb.

Bill Webb.

BW/ee

AUSTRALIAN GRADUATE
SCHOOL OF MANAGEMENT
IN THE
UNIVERSITY OF NEW SOUTH WALES
P.O. BOX 1 • KENSINGTON • NEW SOUTH WALES • AUSTRALIA • 2033
TELEGRAPHIC ADDRESS: UNITECH, SYDNEY • TELEPHONE [redacted] 662 2600

9th December, 1977

Melvin Ferentz
c/- CUNY/UCC
555 West 57 Street
NEW YORK NY 10019

Dear Melvin,

This University has been a UNIX licensee for close to three years and UNIX is now running on eight 11/40s and two 11/70s on this campus. Five of 11/40's function as Batch Stations to a CYBER 72 - the communications is fairly primitive since it uses the USER-200 protocol which is "pre-historic" in origin. I have developed a simple batch system for use on these machines which allows for users to submit (via a cardreader) jobs for the CYBER or local FORTRAN, MACRO or PASCAL jobs. This facility is proving very useful since large numbers of students are taught MACRO at the moment and local turnaround is considerably faster than using the CYBER. Aside from all this the 11/40s are also used for normal timeshare in the best UNIX tradition.

We have made a number of significant changes to UNIX which are of general applicability to all users. PARAM.H details the operating system changes/improvements that have been made.

IN SUMMARY - see param.h for full documentation

i. Full Support for 11/70 CPUS

ii. Shared Data Segments - a group of programs may share a common data segment - done thru a '412' a.out. P and V system calls provide all needed synchronization.

iii. Eliminate all i/0 from EXIT-WAIT sequence by storing post mortem information in the proc entry - uses no more core

iv. Don't wake init every time a process terminates

v. CLKTIM and CPUTIM system calls implemented to provide a unique signal after a given period of elapsed or cpu time

vi. Expand the available address space for the resident operating system on 11/40s. KA5 is used to address device drivers and the routines from SYS?.C. A new linker SYSLD is required

vii. Re-write of timeout routines to be efficient

viii. Implement Stack Limit register

ix. Re-write of sleep system call - no longer grossly inefficient

x. Raw buffer pool used for all raw i/0 (including swapping)

xi. Eliminate all unnecessary swap outs of text segments

xii. Proper handling of write-behind blocks

Changes iii, iv, xii reduced the number of swaps occuring by 90% on my 11/70.

In addition a number of corrections have been made to the distributed code to fix 'bugs'.

These changes are now ready for distribution. Please find enclosed one 2400 ft mag tape containing our system source and miscellaneous support commands and/or files of interest. It would seem inadvisable to distribute all mods because of the enormous problems associated with evaluating same. The tape is recorded at 800 bpi and was produced by a standard TP(MDIRENT=1200.) capable of handling 1200 files.

The system as distributed has been functioning on my School's 11/70 and several 11/40's for some months without problems, although many sections have been in use for longer periods. Recently a new 11/40 (small core) system has experienced problems (a 'crash' every two/three days). At this stage I am unable to ascertain what is causing them. In any event I will notify of any bugs we discover and supply fixes.

You may distribute a copy on the tape to any UNIX licensee of your choice.

Could you please place the THIRD SOFTWARE DISTRIBUTION on the enclosed tape and return it by AIR MAIL or AIR PARCEL POST. Enclosed is a bank draft for US$21.00 to defray your return mailing costs.

Sincerely

I Johnstone

Ian Johnstone

IJ/jc

encl.

UNIVERSITY OF CALIFORNIA, SAN FRANCISCO

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO SANTA BARBARA • SANTA CRUZ

SCHOOL OF PHARMACY
DEPARTMENT OF PHARMACEUTICAL CHEMISTRY SAN FRANCISCO, CALIFORNIA 94143

9 December 1977

Professor Melvin Ferentz
Physics Department
Brooklyn College of CUNY
Brooklyn, New York 11210

The memory management unit in the PDP-11/45 and 11/70 computers offer several advantages over those found in the other PDP-11 family computers. Among the more powerful features is the ability to separate programs into instruction segments and data segments, each segment having the ability to directly reference 64K bytes of memory.

Four PDP-11 instructions facilitate program communication between different addressing modes and instruction/data areas in memory. These are "move to/from previous instruction/data memory space" (mtpi, mfpi, mtpd, mfpd). These instructions are used extensively by UNIX to fetch and store data between user programs and the operating system.

Because of DEC's desire to "...preserve the integrity of proprietary programs", the 'mfpi' instruction does not work correctly when executed with a processor status word equal to 17xxxx (i.e. both current and previous modes are USER). This fact prevents the C subroutine 'nargs.s' from operating as intended when instruction and data space are separated.

There are several solutions to this deficiency, among these are:

1. Implement a system call which executes (correctly) the instruction 'mfpi' with source equal to USER I space.
2. Modify the C compiler to generate an argument count on each and every procedure call.
3. Modify UNIX to use the supervisor mode memory management registers in a way different then it currently does.
4. Modify the hardware to work more "correctly".

The first of these solutions proved unacceptable in terms of overhead time; each system call on the PDP-11/70 consumes equal or greater than 320 microseconds, and 'nargs.s' has the potential for making several of these. In our real time environment this prohibited use of separate I&D space. The second and third solutions were unattractive from a software standards point of view.

After several telephone calls to DEC representatives and a few hours of looking at microcode and logic schematics, we have arrived at a simple modification to (at least) the PDP-11/70 cpu to allow the 'mfpi' instruction to function properly. The modification takes about 15 minutes for an experienced person to implement and involves cutting one foil etch and adding one jumper wire to the M8138-YA memory management board. The relevant section of the PDP-11/70 schematic, DEC #M8138-0-1, "SYS. STATUS REG. (SSRB)", is enclosed. Please note this is as it appears on our KB11-C central processing unit and may very with other processors.

Also included is a listing of our modified 'nargs.s' subroutine that utilizes the "new" instruction.

We have been running under the modified hardware for several weeks now and have had no problems. In particular, absolutely no changes have been made to any other portion of the UNIX system other than the 'nargs.s' routine. On our system the PDP-11/70 memory management diagnostic program, MAINDEC-11-DEKBE-B, was patched to skip around the relevant portion of the diagnostic which tests the "proprietary program" feature; this is optional.

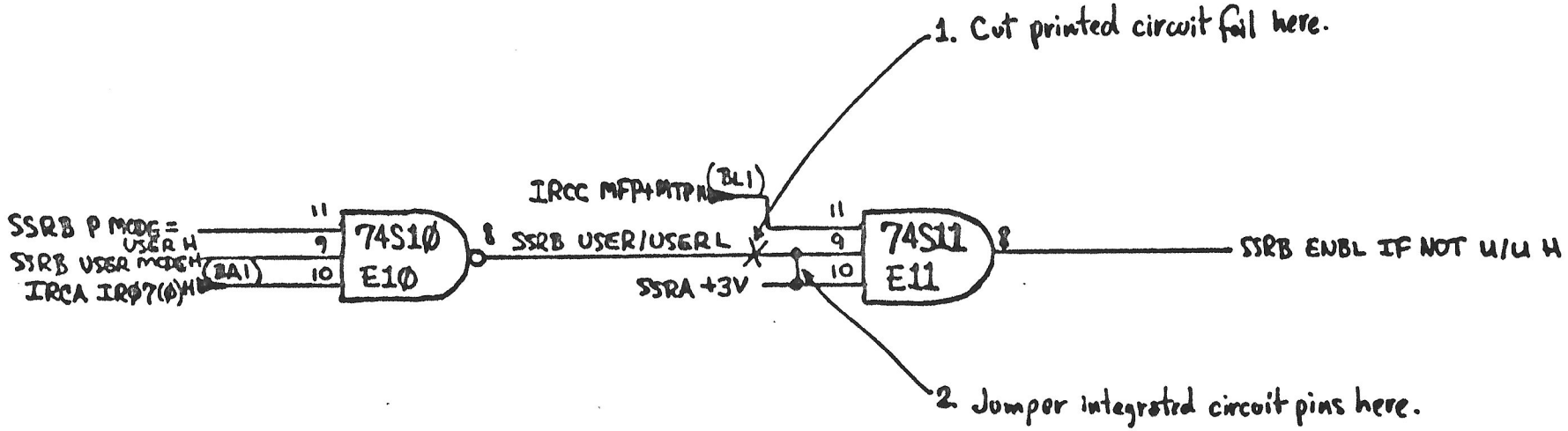
While this solution also has some drawbacks, it has proved quite successful for our application involving real time interactive computer graphics. I trust this information will be useful to others as well.

Sincerely,

Thomas Ferrin

Thomas Ferrin
Computer Facility Manager
Computer Graphics Laboratory

Enclosures



Address Space Control Logic Slot 1

11/7Ø

Title

SYS. STATUS REG.

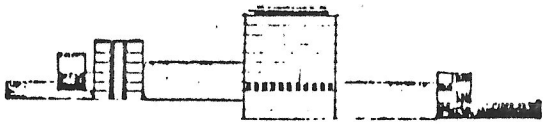
(SSRB)

MB138-Ø-1

```
/ C library -- nargs

.globl  _nargs
mfpi = 6500^tst

_nargs:
        mov     r5,-(sp)
        tstb    initf
        bne     5f
        incb    initf
        clr     r0
        mfpi    (r0)
        cmp     (sp)+,(r0)
        beq     1f
        negb    initf           / separate I&D
        br      4f
1:
        mov     $2,r0
        mfpi    (r0)
        cmp     (sp)+,(r0)
        beq     3f
        negb    initf           / separate I&D
        br      4f
5:
        bit     4f

/* ------------------------     */
3:
        mov     2(r5),r1        / pc of caller of caller
        mov     sp,r5
        clr     r0
        cmp     -4(r1),jsrsd
        bne     8f
        mov     $2,r0
8:
        cmp     (r1),tst1
        bne     1f
        add     $2,r0
        br      2f
1:
        cmp     (r1),cmpi
        bne     1f
        add     $4,r0
        br      2f
1:
        cmp     (r1),addi
        bne     1f
        add     2(r1),r0
        br      2f
1:
        cmp     (r1),jmpi
        bne     1f
        add     2(r1),r1
        add     $4,r1
        br      8b
1:
        cmpb    1(r1),bri+1
        bne     2f
        mov     r0,-(sp)
        mov     (r1),r0
        swab    r0
        ash     $-7,r0
        add     r0,r1
        add     $2,r1
        mov     (sp)+,r0
        br      8b
2:
```

Original 'nargs'


```
        asr     r0
        mov     (sp)+,r5
        rts     pc

/* ------------------------     */
4:
        mov     2(r5),r1
        mov     sp,r5
        clr     r0
        mfpi    -4(r1)
        cmp     (sp)+,jsrsd
        bne     8f
        mov     $2,r0
8:
        mfpi    (r1)
        cmp     (sp),tst1
        bne     1f
        add     $2,r0
        br      2f
1:
        cmp     (sp),cmpi
        bne     1f
        add     $4,r0
        br      2f
1:
        cmp     (sp),addi
        bne     1f
        tst     (sp)+
        mfpi    2(r1)
        add     (sp),r0
        br      2f
1:
        cmp     (sp),jmpi
        bne     1f
        tst     (sp)+
        mfpi    2(r1)
        add     (sp)+,r1
        add     $4,r1
        br      8b
1:
        cmpb    1(sp),bri+1
        bne     2f
        mov     r0,-(sp)
        mov     2(sp),r0
        swab    r0
        ash     $-7,r0
        add     r0,r1
        add     $2,r1
        mov     (sp)+,r0
        tst     (sp)+
        br      8b
2:
        tst     (sp)+
        asr     r0
        mov     (sp)+,r5
        rts     pc

.data
jsrsd:  jsr     pc,*$0
tst1:   tst     (sp)+
cmpi:   cmp     (sp)+,(sp)+
addi:   add     $0,sp
jmpi:   jmp     0
bri:    br      .
initf:  .=.+1   / 0 == not initialized
                / 1 == initialized
                / 377 == init + I&D separated
```



בית הספר לרפואה של האוניברסיטה העברית והדסה · ירושלים
THE HEBREW UNIVERSITY-HADASSAH MEDICAL SCHOOL · JERUSALEM
Experimental Medicine & Cancer Research

ת. ד. 1172-טלפון 426111
כא חייג ישיר 428
P.O.B. 1172 — TEL. 426111
מיקוד 91-000 ירושלים

E/Md

December 14, 1977.

Professor M. Ferentz,
CUNY/UCC,
555 West 57 Street,
New York, N.Y. 10019,
U.S.A.

Dear Professor Ferentz:

The Computer Science Laboratory of our Institute of Mathematics suggested I turn to you for help in finding a continuous system modelling programme for use with our PDP11/03 (24K, 2 diskettes, extended arithmetic chip for both fixed and floating point, VT55 graphics, teletype) under UNIX.

Thank you for this courtesy.

Yours sincerely,

Norman Grover

Professor N. B. Grover.

UNIVERSITY OF WASHINGTON
SEATTLE, WASHINGTON 98195

Department of Computer Science, FR-35
(Area Code 206) 543- 2697

December 19, 1977

Professor Melvin Ferentz
Physics Department
Brooklyn College of CUNY
Brooklyn, NY 11210

Dear Professor Ferentz,

I have developed a system that permits up to 81 UNIX processes to send information to any other process in the system using only 2 of the file descriptors allowed per UNIX process. I, like E.G. Keizer (UNIX NEWS 2, 5 1977), had problems with intermixed messages when more than one process writes to the same pipe. Enclosed is a listing of a modified system pipe read/write routine that corrects the problem.

I have also found it useful to be able to obtain certain information about a pipe. Does a file descriptor refer to a pipe? Is it the read or write end? How many characters are in a pipe? Do two file descriptors refer to opposite ends of the same pipe? A system call pstat has been added to provide this information. Enclosed are listings of the necessary routines and a manual for the system call.

Do you know of anyone who has implemented pipes as true circular buffers?

Sincerely,

Daniel Lipkie

Daniel Lipkie

Enclosures: pipe.c
pstat.s
pstat(II)

DL/kh

```
#
/*
 */

#include "../param.h"
#include "../systm.h"
#include "../user.h"
#include "../inode.h"
#include "../file.h"
#include "../reg.h"

/*
 * Max allowable buffering per pipe.
 * This is also the max size of the
 * file created to implement the pipe.
 * If this size is bigger than 4096,
 * pipes will be implemented in LARG
 * files, which is probably not good.
 */
#define PIPSIZ  4096

/*
 * The sys-pipe entry.
 * Allocate an inode on the root device.
 * Allocate 2 file structures.
 * Put it all together with flags.
 */
pipe()
{
        register *ip, *rf, *wf;
        int r;

        ip = ialloc(rootdev);
        if(ip == NULL)
                return;
        rf = falloc();
        if(rf == NULL) {
                iput(ip);
                return;
        }
        r = u.u_ar0[R0];
        wf = falloc();
        if(wf == NULL) {
                rf->f_count = 0;
                u.u_ofile[r] = NULL;
                iput(ip);
                return;
        }
        u.u_ar0[R1] = u.u_ar0[R0];
        u.u_ar0[R0] = r;
        wf->f_flag = FWRITE|FPIPE;
        wf->f_inode = ip;
        rf->f_flag = FREAD|FPIPE;
        rf->f_inode = ip;
        ip->i_count = 2;
        ip->i_flag = IACC|IUPD;
        ip->i_mode = IALLOC;
}
```



```
/*
 * Read call directed to a pipe.
 */
readp(fp)
int *fp;
{
        register *rp, *ip;

        rp = fp;
        ip = rp->f_inode;

loop:
        /*
         * Very conservative locking.
         */
        plock(ip);

        /*
         * If the head (read) has caught up with
         * the tail (write), reset both to 0.
         */
        if(rp->f_offset[1] == ip->i_size1) {
                if(rp->f_offset[1] != 0) {
                        rp->f_offset[1] = 0;
                        ip->i_size1 = 0;
                        if(ip->i_mode&IWRITE) {
                                /* removed 12 Dec 77 ***
                                *  ip->i_mode =& ~IWRITE;
                                *** end removal */
                                wakeup(ip+1);
                        }
                }

                /*
                 * If there are not both reader and
                 * writer active, return without
                 * satisfying read.
                 */
                prele(ip);
                if(ip->i_count < 2)
                        return;
                ip->i_mode =| IREAD;
                sleep(ip+2, PPIPE);
                goto loop;
        }

        /*
         * Read and return
         */
```

(a)

```
        u.u_offset[0] = 0;
        u.u_offset[1] = rp->f_offset[1];
        readi(ip);
        rp->f_offset[1] = u.u_offset[1];
        prele(ip);
}

/*
 * Write call directed to a pipe.
 */
writep(fp)
{
        register *rp, *ip, c;

        rp = fp;
        ip = rp->f_inode;
        c = u.u_count;

        /*
         * Only one writer at a time through here
         */
loop1:
        plock(ip);
        while(ip->i_mode&IWRITE) {
                rp->i_flag =| IOTHER;
                prele(ip);
                sleep(ip+3, PPIPE);
                goto loop1;
        }
        ip->i_mode =| IWRITE;
        goto loop2;

        /*
         * If all done, return.
         */

loop:
        plock(ip);
loop2:
        if(c == 0) {
                ip->i_mode =& ~IWRITE;
                if(rp->i_flag&IOTHER) {
                        rp->i_flag =& ~IOTHER;
                        prele(ip);
                        wakeup(ip+3);
                }
                else prele(ip);
                u.u_count = 0;
                return;
        }

        /*
         * If there are not both read and
         * write sides of the pipe active,
         * return error and signal too.
         */
```



```
        if(ip->i_count < 2) {
                prele(ip);
                u.u_error = EPIPE;
                psignal(u.u_procp, SIGPIPE);
                return;
        }

        /*
         * If the pipe is full,
         * wait for reads to deplete
         * and truncate it.
         */

        if(ip->i_size1 == PIPSIZ) {
                prele(ip);
                sleep(ip+1, PPIPE);
                goto loop;
        }

        /*
         * Write what is possible and
         * loop back.
         */

        u.u_offset[0] = 0;
        u.u_offset[1] = ip->i_size1;
        u.u_count = min(c, PIPSIZ-u.u_offset[1]);
        c -= u.u_count;
        writei(ip);
        prele(ip);
        if(ip->i_mode&IREAD) {
                ip->i_mode =& ~IREAD;
                wakeup(ip+2);
        }
        goto loop;
}

/*
 * Get pipe status
 */
pstat() {
        register up,*fp,*ip;
        u.u_ar0[R0] = 0; /* u.u_ar0[R0] used for temporary storage */
        up = u.u_arg[0];
        if(fuword(up) < 0) goto pstat0; /* -1 means no file descriptor */
        fp = getf(fuword(up));
        if(fp == NULL) {
                u.u_error = -1;  goto pstat0; }
        if(!(fp->f_flag&FREAD && fp->f_flag&FPIPE)) {
                u.u_error = -2;  goto pstat0; }
        ip = fp->f_inode;
        u.u_ar0[R0] = ip;
        plock(ip);
        suword(up+6,ip->i_size1);
        suword(up+4,fp->f_offset[1]);
```

```
pstat0:
        if(fuword(up+2) < 0) goto pstat1;
        fp = getf(fuword(up+2));
        if(fp == NULL) {
                u.u_error = -3;  goto pstat1; }
        if(!(fp->f_flag&FWRITE && fp->f_flag&FPIPE)) {
                u.u_error = -4;  goto pstat1; }
        if(u.u_ar0[R0] == 0) {
                ip = fp->f_inode;
                u.u_ar0[R0] = ip;
                plock(ip);
                suword(up+6,ip->i_size1);
        }
        else if(ip != (fp->f_inode>>1) ) {
                u.u_error = -5;  goto pstat1; }
pstat1:
        suword(up+8,u.u_error);
        if(u.u_error < 0) u.u_error = EBADF;
        if(u.u_ar0[R0] != 0) {
                prele(ip);
                u.u_ar0[R0] = 0;
        }
}

/*
 * Lock a pipe.
 * If its already locked,
 * set the WANT bit and sleep.
 */
plock(ip)
int *ip;
{
        register *rp;

        rp = ip;
        while(rp->i_flag&ILOCK) {
                rp->i_flag =| IWANT;
                sleep(rp, PPIPE);
        }
        rp->i_flag =| ILOCK;
}

/*
 * Unlock a pipe.
 * If WANT bit is on,
 * wakeup.
 * This routine is also used
 * to unlock inodes in general.
 */
prele(ip)
int *ip;
{
        register *rp;

        rp = ip;
        rp->i_flag =& ~ILOCK;
```



```
        if(rp->i_flag&IWANT) {
                rp->i_flag =& ~IWANT;
                wakeup(rp);
        }
}
```

ALSO

add a flag bit to

/usr/source/sys/ken/inode.h

#define IOTHER 0

```
/
/ pstat(buf);   16 Dec 77       (Daniel Lipkie)
/ struct pstatus *buf;
/
.globl  _pstat, cerror
pstat   = 40.

_pstat:
        mov     r5,-(sp)
        mov     sp,r5
        mov     4(r5),0f
        sys     0; 9f
        bec     1f
        jmp     cerror
1:
        mov     (sp)+,r5
        rts     pc
.data
9:
        sys     pstat;  0:..
```

ALSO

change /~~usr~~/~~sys~~/
/usr/source/sys/ken/sysent.c

1, &pstat /* 40 = pstat */

NAME

pstat - get pipe status

SYNOPSIS

```
(pstat = 40.)
sys pstat; buf

pstat(buf);
struct pstatus *buf;
```

DESCRIPTION

Buf is the address of a 5 word buffer into which information is placed concerning the pipe. A pipe is implemented as a fixed-length non-circular buffer of 4096 bytes. Two file descriptors for use in read and write operations are associated with a pipe (pipe (II)). Two indices (0 - 4096) are used to indicate the next character to be returned by a read or filled by a write. When the indices are equal a read will reset both to 0 and suspend the process awaiting a write. When the indices are equal a write will add to the buffer starting at the current index. When the write index reaches 4096 the writing process is suspended until read opeations empty the pipe and a successive read resets the indices. The pstat system call uses the following structure:

```
struct pstat {
  int   fildes[2];      /* [0] - the read fildes or -1 */
                        /* [1] - the write fildes or -1 */
  int   r_index;        /* read index */
  int   w_index;        /* write index */
  int   cmpl_code;      /* completion code */
};
```

The pipe read and/or write file descriptors are passed to pstat which returns the indices and a completion code. -1 is used to indicate that the corresponding file descriptor is not being passed. If a read file descriptor is passed both indices are returned. If only the write file descriptor is passed only the write index is returned. A completion code is always returned and has the following meaning:

```
  0 -  normal return
 -1 -  fildes[0] not a valid file descriptor
 -2 -  fildes[0] not the read end of a pipe
 -3 -  fildes[1] not a valid file descriptor
 -4 -  fildes[1] not the write end of a pipe
 -5 -  fildes are not opposite ends of same pipe
```

SEE ALSO

pipe(II), ls (I), fstat (II), fs (V)

DIAGNOSTICS

Error bit (c-bit) is set if the completion code is < 0. From C, a -1 is returned if the completion code is < 0 and errno is set to 9.

Rochester Institute of Technology

School of Computer Science and Technology

One Lomb Memorial Drive
Rochester New York 14623
716-464-2995

December 21, 1977

Professor Melvin Ferentz
Department of Physics
Brooklyn College of CUNY
Brooklyn, N. Y. 11210

Dear Professor Ferentz:

A check for $20.00 has been sent to you for subscriptions to UNIX News, Volumes 1 and 2. I understand that Volume 1 will be sent upon receipt of payment since we only received issues 1-8 of Volume 1.

Having received only Volume 2, however, I feel like a person who has joined a discussion which has been in progress for some time. Many of the articles and announcements are very intriguing, such as the hotline and items on the distribution tape, but I know I am missing some essential details. Perhaps Volume 1 contains the UNIX Rosetta Stone.

I have noticed that several inquiries for floppy disk drivers have appeared in UNIX News. We have developed an RX11 driver as a matter of necessity, since we do not have magnetic tape backup. The driver depends on the fact that the buffers are in the kernel address space, so it's not as general as one might like. It can support a file system, though, and that is sufficient for our purposes. We can only produce machine readable information on RK05 disks and RX11 diskettes, which is probably useless, but we are willing to distribute the listing.

We think there is a bug in the use of mountable devices. The following sequence will result in the i-node for "directory" having # links to it but a reference count of 1:

```
/etc/mkfs  /dev/whatever ...
mkdir  directory
/etc/mount  /dev/whatever  directory
rmdir  directory
/etc/umount  /dev/whatever
```

We just discovered this the other day and we have not had time to track down the cause, but we thought this might be informative to others.

Yours truly,

Michael J. Lutz

MJL:ba

UNIVERSITY OF CALIFORNIA, SANTA BARBARA

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO

SANTA BARBARA • SANTA CRUZ

COMPUTER CENTER

SANTA BARBARA, CALIFORNIA 93106

January 11, 1978

Melvin Ferentz
c/o CUNY/UCC
555 West 57th Street
New York, New York 10019

Dear Mel,

Three items which may be of interest to the Unix community:

We have found a dramatic improvement in performance on our PDP 11/45 through the installation of 2K cache memory (ours is from Able Computer Technology). Installations trying to get more horsepower for their machines should investigate this alternative.

We have been running a System Industries controller driving a CDC 300 meg disk drive for about four months and are very pleased with reliability and performance. We have had no hardware or software problems with this disk system which was operational within one day after installation.

We have telecommunications between our IBM 360/75 under HASP-OS/MVT and Unix as well as multiplexing of 16 virtual tty's on one physical dh11 line. Both of these features utilize part of an Interdata Model 50 with locally written software. We would be interested in dialogue with other installations regarding these facilities.

It is about 70 degrees and sunny in Santa Barbara.

Dwight W. McCann

Dwight McCann
Facilities Support Programmer
Computer Center

DM:bd

THE UNIVERSITY OF NEW SOUTH WALES

P.O. BOX 1 • KENSINGTON • NEW SOUTH WALES • AUSTRALIA • 2033
TELEPHONE 663 0351
EXTN.

PLEASE QUOTE

COMPUTING SERVICES UNIT

CW:jmg

78/01/16

Professor Melvin Ferentz,
c/- CUNY/UCC,
555 West 57 Street,
NEW YORK, N.Y. 10019, U.S.A.

Dear Professor Ferentz,

Our unit operates a CDC Cyber under KRONOS, so we have developed a Source Library maintenance program, SLUP, similar to the CDC MODIFY utility, for our UNIX software.

The program is a line-oriented editor which holds source modules together with sets of modifications in one library file. The major virtue of the program is that it allows reconstruction of the several stages of a module, from the original unmodified source to the current version. Comment facilities allow one to record with the modification sets such information as date amended, why, and by whom, as well as the actual changes.

We can supply source and user documentation on a 600 ft. tape for $30 nett, or for $15 plus postage if a tape is sent to us.

Sincerely,

Colin Webb

COLIN WEBB,
Senior Analyst Programmer

WASHINGTON STATE UNIVERSITY
PULLMAN, WASHINGTON 99163

COMPUTER SCIENCE DEPARTMENT

29 January 1978

Bruce R. Hilditch
Computer Science Dept.
Washington State University
Pullman, Washington 99163

Melvin Ferentz
c/o CUNY/UCC
555 West 57th Street
New York, NY 10019

Dear Mr. Ferentz,

Can you (and would you?) kindly give me the address/telephone number of a member of the Unix User's Group that can supply me with a copy of PASCAL suitable for use under UNIX running on a PDP 11/60? We will naturally pay for any nominal costs for copying and shipping and so forth. Almost any hard copy format will be acceptable.

Your assistance will be greatly appreciated.

Thank you,

Bruce R. Hilditch

UNIVERSITY OF CALIFORNIA, SAN FRANCISCO

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO — SANTA BARBARA • SANTA CRUZ

COMPUTER CENTER

SAN FRANCISCO, CALIFORNIA 94143

January 18, 1978

Melvin Ferentz
c/o Cuny/UCC
555 W. 57th Street
New York, NY 10019

Dear Prof. Ferentz:

We are currently investigating methods for implementing data entry applications (Key-to-What?) at UCSF. Since UNIX is interesting to our community for other reasons anyway, we are led to ask whether anyone has written data entry software under UNIX. In particular, does anyone have experience with any UNIX based system which makes it easy to validate data as is is entered? The validation criteria which interest us include both "must be numeric" and "must be in this ten megabyte file of valid accounts".

Please write: R. H. Karpinski
U-76 UCSF
San Francisco, CA 94143

or call collect: 415-666-4529
if you have relevant information.

Sincerely yours,

Richard H. Karpinski

University of Essex

Department of Electrical
Engineering Science
Wivenhoe Park
Colchester CO4 3SQ

Tel: Colchester 44144 (STD Code 020 6)
Telegraphic address: University Colchester
Telex: 98440 (UNILIB COLCHSTR)

17 January 1978

Professor Ferentz
c/o CUNY/UCC
555 West 57 Street
New York
NY 10019
USA

Dear Professor Ferentz

1 Could you send us a copy of the "diff listing of changes/bug fixes to UNIX" which I've seen mentioned a couple of times in the Unix Newsletter/";login:".

2 We have a DZ driver (smaller than the Nijmegen one in the third distribution but not as good and no modem control) and a DR-11C driver (seems to work, but anyway it's a reasonable place to start!) if anyone wants them.

3 Pete Madams here developed a "Thrust Meter" which shows Bus Busy (as a percentage) on a beautiful old 9" meter which is labelled THRUST LBS x 1000 and is calibrated from 0 to 10. It's so useful as an indication of system loading that several users remote from the machine take the signal and have smaller meters in their offices! It doesn't work with DEC operating systems as they don't use the WAIT instruction but idle in a low-priority loop. The drawing should reproduce OK if it's done full-size.

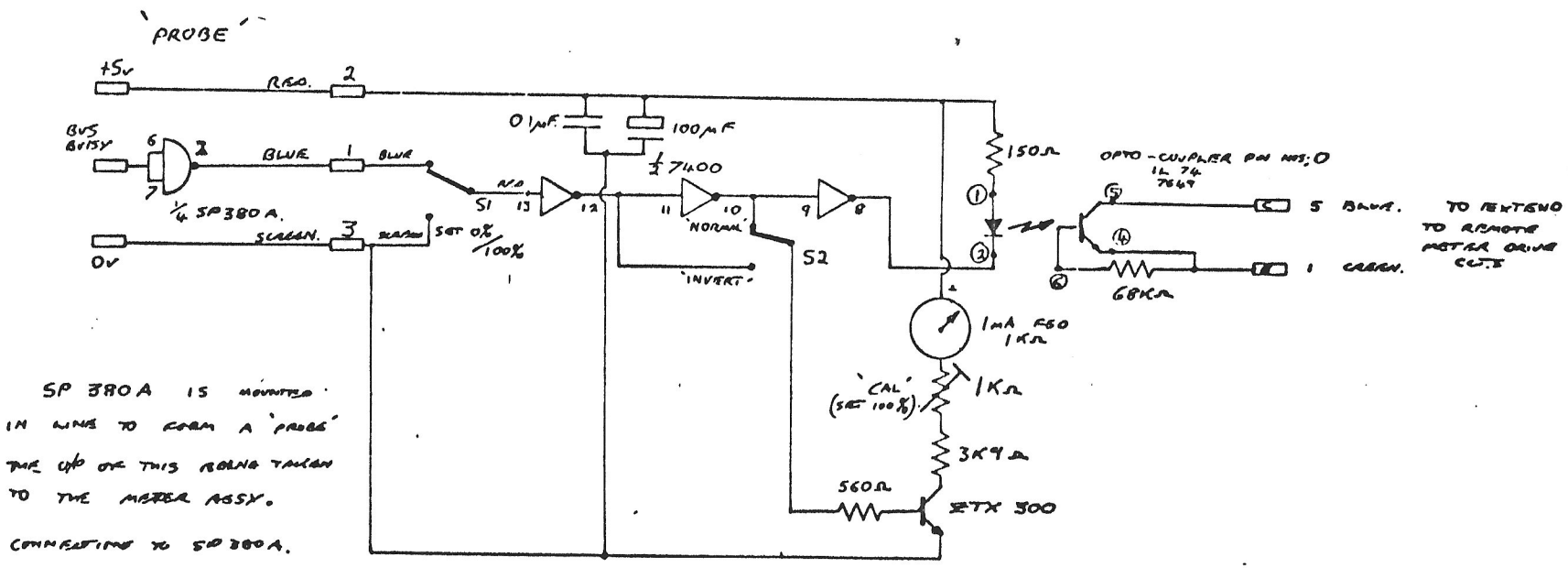
Keep up the good work with ";login:".

Yours sincerely

Bruce Anderson

D B Anderson

'THRUST' METER CCT.



SP 380A IS MOUNTED IN LINE TO FORM A 'PROBE' THE O/P OF THIS BOARD TAKEN TO THE METER ASSY.

CONNECTIONS TO SP 380A.

1, 4, 5, 9, 10, 11, 12 } 0v
8 +5v
3, 13, 14 } NC
2, 6, 7 } AS ABOVE.

TO CALIBRATE; ENSURE CAL POT IS AT MAX RESISTANCE. SET S1 TO GROUND; S2 TO 'NORMAL': METER SHOULD ZERO ON SWITCH ON. IF NOT CHECK MECHANICAL STOP. NOW SET S2 TO 'INVERT' AND ADJUST CAL POT UNTIL METER READS 100% (F.S.D). RESET SWITCHES TO NORMAL WORKING CONDITIONS i.e. S1 TO 'PROBE' & S2 TO 'NORMAL'

DESIGN;
P.H. MADAMS
OPTO COUPLED EXTENDER
C.C. RUMBOLD.

ORIGIN: 7/77
MODS 1/9/77 (RMSOO)