UNIX version 7 bugs

This document describes the UNIX version 7 errors fixed at the Vrije Universiteit, Amsterdam. Several of these are discovered at the VU. Others are quoted from a list of bugs distributed by BellLabs.

For each error the differences between the original and modified source files are given, as well as a test program.

ERROR 1: C optimizer bug for unsigned comparison

The following C program caused an IOT trap, while it should not (compile with 'cc -O prog.c'):

```
unsigned        i = 0;

main() {
        register j;

        j = -1;
        if (i > 40000)
                abort();
}
```

BellLabs suggests to make the following patch in c21.c:

```
/* modified /usr/src/cmd/c/c21.c */

189             if (r==0) {
190     /* next 2 lines replaced as indicated by
191      * Bell Labs bug distribution ( v7optbug )
192                     p->back->back->forw = p->forw;
193                     p->forw->back = p->back->back;
194       End of lines changed */
195                     if (p->forw->op==CBR
196                       || p->forw->op==SXT
197                       || p->forw->op==CFCC) {
198                             p->back->forw = p->forw;
199                             p->forw->back = p->back;
200                     } else {
201                             p->back->back->forw = p->forw;
202                             p->forw->back = p->back->back;
203                     }
204     /* End of new lines */
205                     decref(p->ref);
206                     p = p->back->back;
207                     nchange++;
208             } else if (r>0) {
```

Use the previous program to test before and after the modification.

ERROR 2: The loader fails for large data or text portions

The loader ’ld’ produces a "local symbol botch" error for the following C program.

```
int     big1[10000] = {
        1
};
int     big2[10000] = {
        2
};

main() {
        printf("loader is fine\n");
}
```

We have made the following fix:

```
/* original /usr/src/cmd/ld.c */

113     struct {
114             int     fmagic;
115             int     tsize;
116             int     dsize;
117             int     bsize;
118             int     ssize;
119             int     entry;
120             int     pad;
121             int     relflg;
122     } filhdr;
```

```
/* modified /usr/src/cmd/ld.c */

113     /*
114      * The original Version 7 loader had problems loading large
115      * text or data portions.
116      * Why not include <a.out.h> ???
117      * then they would be declared unsigned
118      */
119     struct {
120             int     fmagic;
121             unsigned        tsize;          /* not int !!! */
122             unsigned        dsize;          /* not int !!! */
123             unsigned        bsize;          /* not int !!! */
124             unsigned        ssize;          /* not int !!! */
125             unsigned        entry;          /* not int !!! */
126             unsigned        pad;            /* not int !!! */
127             unsigned        relflg;         /* not int !!! */
128     } filhdr;
```

ERROR 3: Floating point registers

When a program is swapped to disk if it needs more memory, then the floating point registers were not saved, so that it may have different registers when it is restarted. A small assembly program demonstrates this for the status register. If the error is not fixed, then the program generates an IOT error. A "memory fault" is generated if all is fine.

```
start:  ldfps   $7400
1:      stfps   r0
        mov     r0,-(sp)
        cmp     r0,$7400
        beq     1b
        4
```

Some digging into the kernel is required to fix it. The following patch will do:

```
/* original /usr/sys/sys/slp.c */

                a2 = malloc(coremap, newsize);
                if(a2 == NULL) {
                        xswap(p, 1, n);
                        p->p_flag |= SSWAP;
                        qswtch();
                        /* no return */
                }

/* modified /usr/sys/sys/slp.c */

                a2 = malloc(coremap, newsize);
                if(a2 == NULL) {
        #ifdef FPBUG
                        /*
                         * copy floating point register and status,
                         * but only if you must switch processes
                         */
                        if(u.u_fpsaved == 0) {
                                savfp(&u.u_fps);
                                u.u_fpsaved = 1;
                        }
        #endif
                        xswap(p, 1, n);
                        p->p_flag |= SSWAP;
                        qswtch();
                        /* no return */
                }
```

ERROR 4: Floating point registers.

A similar problem arises when a process forks. The child will have random floating point registers as is demonstrated by the following assembly language program. The child process will die by an IOT trap and the father prints the message "child failed".

```
exit    = 1.
fork    = 2.
write   = 4.
wait    = 7.

start:  ldfps   $7400
        sys     fork
        br      child
        sys     wait
        tst     r1
        bne     bad
        stfps   r2
        cmp     r2,$7400
        beq     start
        4
child:  stfps   r2
        cmp     r2,$7400
        beq     ex
        4
bad:    clr     r0
        sys     write;mess;13.
ex:     clr     r0
        sys     exit

        .data
mess:   <child failed\n>
```

The same file slp.c should be patched as follows:

```
/* original /usr/sys/sys/slp.c */

499             /*
500              * When the resume is executed for the new process,
501              * here's where it will resume.
502              */
503             if (save(u.u_ssav)) {
504                     sureg();
505                     return(1);
506             }
507             a2 = malloc(coremap, n);
508             /*
509              * If there is not enough core for the
510              * new process, swap out the current process to generate the
511              * copy.
512              */

/* modified /usr/sys/sys/slp.c */
```

```
	/*
	 * When the resume is executed for the new process,
	 * here's where it will resume.
	 */
	if (save(u.u_ssav)) {
		sureg();
		return(1);
	}
#ifdef FPBUG
	/* copy the floating point registers and status to child */
	if(u.u_fpsaved == 0) {
		savfp(&u.u_fps);
		u.u_fpsaved = 1;
	}
#endif
	a2 = malloc(coremap, n);
	/*
	 * If there is not enough core for the
	 * new process, swap out the current process to generate the
	 * copy.
	 */
```

ERROR 5: /usr/src/libc/v6/stat.c

Some system calls are changed from version 6 to version 7. A library of system call entries, that make a version 6 UNIX look like a version 7 system, is provided to run some useful version 7 utilities, like 'tar', on UNIX-6. The entry for 'stat' contained two bugs: the 24-bit file size was incorrectly converted to 32 bits (sign extension of bit 15) and the uid/gid fields suffered from sign extension.

Transferring files from version 6 to version 7 using 'tar' will fail for all files for which

```
( (size & 0100000) != 0 )
```

These two errors are fixed if stat.c is modified as follows:

```
/* original /usr/src/libc/v6/stat.c */

	char  os_size0;
	short os_size1;
	short os_addr[8];

	buf->st_nlink = osbuf.os_nlinks;
	buf->st_uid = osbuf.os_uid;
	buf->st_gid = osbuf.os_gid;
	buf->st_rdev = 0;

/* modified /usr/src/libc/v6/stat.c */

	char  os_size0;
	unsigned os_size1;
	short os_addr[8];
```

```
        buf->st_nlink = osbuf.os_nlinks;
        buf->st_uid = osbuf.os_uid & 0377;
        buf->st_gid = osbuf.os_gid & 0377;
        buf->st_rdev = 0;
```