November 1982 Vol IV No VI

Australian UNIX Users Group NEWSLETTER

[This month I have handed over responsibility for AUUGN content to a guest editor. So don't blame me! Bob Kummerfeld]

Hello, I'm Karlos Mauvtaque, UNIX[1] enthusiast, seventh Dan rogue player, four times total winner and founding member of the Australian Unix Red Wine Users Group (AURWUG). And I'm your user-friendly menu-driven guest editor.

Well '82 has sure turned out to be a good year. Those keen lads at UCB announced 4.2bsd, destined to change the way the disks spin (remember the secret is to bang the rocks together guys). Innovation in operating system design and implementation has long been a research interest of mine and included in this issue is my analysis of what lies in writable-control-store for us in 4.3bsd.

'82 has also been a great year for vendors. Many interesting non-commercial sales pitches were seen this year at AUUGMs. It seems that the dawn of engineering excellence is upon us. Zillions of UNIX look-a-like, smell-a-like and C-a-like systems are upon us. Onyx, Unos, Unison, Zenix, and Buttix, all of them were there in force; crashing merrily and scribbling on their respective file-systems. All of them solving the fundamental UNIX short-coming - a poor choice of command names like "rm" and "dsw".

'82 has also been a great year for the absurd. There was that absurd Californian who taught us all about paging page table entries, and how we'll all make a lot of money selling memory or something. There was the absurd in-place file system conversion suggestion from Mr Level-6.92. There was the absurd tee-shirt bonanza from a leading "buy our machines" vendor. And there was the Red Capsicum incident. Well enough said about that.

What lies ahead of us? Read on...

```
*****************************************************************
*                                                               *
*       S U B S C R I P T I O N   F E E S   N O W   D U E       *
*                                                               *
*       This will be your last issue of AUUGN unless you        *
*       send $24 to the address shown on the back cover.        *
*                                                               *
*       Support the Unix community - share your experiences.    *
*           Send an article about what YOU are doing.           *
*                                                               *
*****************************************************************
```

1 UNIX is a trademark of Bell Laboratories.

[This article was inherited from Chris Rowles who has defected to PSYROMAHHT. No responsibility is assumed for it. It was written in pigeon ms and wouldn't compile let alone run. I have beaten it into shape and chased after a title and author. The current belief is that it appeared in EUUGN last year. - KM]

An Evaluation of ONYX

Alan Mason

Introduction

Thanks to the cooperation of Scan Computer Systems Ltd., an Onyx C8002 system was installed in the Edinburgh University Architecture Department, alongside the existing PDP11/60 installation, for 2 days. Members of the EdLUUG were given free access to the system, and this is the result of their experiences. Many thanks are due to Joel Abramson and Phred Groves, of Scan, for their patient tolerance of the activities of the assembled throng.

Hardware

The system supplied had 512K bytes of main memory, and a single IMI 8-inch winchester disk, of 18M bytes formatted capacity. This hardware costs in the region of 11K pounds. Scan promise upgrades of 1M bytes of main memory, and to 40M and 100M bytes (unformatted) of disk, both to be installed in the same box.

The hardware was installed by plugging it in, attaching the terminals, and turning it on. The only thing to disturb its equanimity for the 2 days was the application of large doses of static, which interrupted the kernel. For most of the time, it ran with the cover lifted, though we were assured that this was purely to satisfy the curiosity of the on-lookers, not to cure heating problems. The relatively small room, in which was installed the Onyx system, an LSI11, an Apple, 7 terminals, and many people, got uncomfortably hot without causing problems.

Kernel

The licence for the software supplied costs in the region of 2K pounds (for 8 users). The kernel appeared, as far as we could determine, to implement all the system calls of V7 UNIX[1] correctly[2], plus a few others. Apart from the static interruptions, the kernel ran reliably. It is relatively large (98K bytes), but this still leaves 414K bytes for user programs.

The kernel comes as a single file, "onix.o", together with an embryo library for extra, user-specific device drivers. The system ".h" files, including "param.h" are available, but changing them won't have much effect! However, the disk layout is declared in "c.c", so this can be changed. Much aggravation was caused by the Bell teletype driver, particularly because of its lack of paging, and inability to erase characters correctly from the VDU screen. An attempt will

1 UNIX is a Trademark of Bell Laboratories.
2 Except "sysphys" and "syslock", neither of which are widely used.

be made to persuade Onyx to supply the EUUG driver as an alternative, or to supply the system as libraries.

Stack growth is not performed by the hardware intercepting and backing-up segmentation violations, but by a system call "ugrow". The "csave" code checks for segment boundary transgression, and calls "ugrow" as appropriate. All segmentation violations are signalled. If "ugrow" fails to obtain more stack, the process concerned receives signal SIGSEGV as expected. Thus, there are no "backup()" problems, provided that your language compiler understands the convention. However, the extra computation is an overhead on "csave" for all function calls, which is probably a significant drag on the overall performance.

Floating point support was so slow that is was initially assumed that it was being dony by software. Typical floating-point add times were 0.3ms, and 0.45ms for double-length. However, we have since been told that the system does use the floating-point hardware, but the overhead of synchronising the two processors is sufficient to account for the timings.

<u>Utilities</u>

init — As usual with Bell V7 systems, the system arrives after a boot with a root shell in single-user mode. When the console logs out, it then executes "/etc/rc" and goes multi-user. Of course, this effectively nullifies all UNIX's file protection and security features. Anyone who can obtain access to the console and the power switch can access, and destroy, any file. Although our systems have been modified to overcome this problem, applying this solution to systems without a removable disk might be risky. It was reputed to be possible to boot a system from the cassette tape, but Scan had encountered some problems in this area.

C — The machine had two C compilers. The earlier compiler, called "cc", appeared to be a descendant of the Ritchie PDP11 compiler. It appeared to implement the language defined in the C book but did not implement structure assignments. It did not support the data types "float" or "double" at all, failing with mysterious messages such as

```
0: Intermediate File Error
```

when values of these types were declared. The code it produced seemed to be about 70% bigger than code compiled with "cc -O" on a PDP11. It even has the token "pdp11" defined! It sometimes dumped core when compiling standard C programs.

The more modern compiler "zcc" did implement structure assignment and floating point arithmetic, but excluded bit fields and static function, both of which are in the C book specification. The code it generated was about 20-25% bigger than the corresponding code from "cc -O" on a PDP11. This improvement may have been due to the optimiser with which it was provided. The object modules it produced always had instruction and data spaces separate. The pre-processor for "zcc" was incompatible with that for "cc", particularly as regards comments on "#define" lines. In general, it seemed even less reliable than "cc", being obviously in an early stage of development. It was inclined to expire in a cloud of error messages, obviously intended more to debug the compiler than to debug the program. Even more seriously, it sometimes appeared to compile programs which then failed to execute correctly.

We have since been told that its author is Dennis Allison, of Stanford.

adb	Adb was capable of symbolic output as Z8000 assembler, except that the system calls were wrongly interpreted, but the documentation still described the PDP11 version. There was no documentation whatsoever on the Z8000 assembler, and Onyx appear reluctant to provide it, even claiming that they don't have it themselves! It is difficult to see how "adb" can be useful without this information.
nroff	Nroff worked on fairly large documents using the "-ms" macros, and the "tbl" preprocessor. However, the terminal driving tables were the standard Bell set, not very useful for normal UK purchasers of daisy-wheel printers. Of course, without the source code it is a bit difficult to construct your own.
yacc	Yacc worked on a single, fairly small, grammar.
lex	Lex worked on a single, very small, grammar.
make	Make worked on a fairly complex "makefile". However, it revealed a problem with "touch" which always returns a failure error code.
pas	This UCSD Pascal compiler proved reasonably satisfactory. It includes ISAM support, which is linked into the user program irrespective of whether it is used. With separate I/D space in use, the ISAM support moves to I-space and leaves about 60K bytes for user programs and data. Without this option, only 48K bytes are available. The following incompatibilities were noted:-

- The user must explicitly "stty cbreak" to enable character level I/O to function as UCSD expects; this should be done by the run-time support.

- The KEYBOARD device is connected to the UNIX file "stderr". This does not make sense, since KEYBOARD is an input device and "stderr" is an output device. Programs using KEYBOARD do not work at present.

- The CLOSE-CRUNCH option is not implelmented, but in addition the CLOSE-PURGE option fails to remove the file. Opening a file with a specific length, as "FILE.TEXT[10]" includes the length specifier in the file name.

- The include pragma {$I...} does not appear to find the include files even when the full path name is supplied.

- The use of segment procedures had no effect on the available space for user programs; they must be held in memory rather than swapped on demand.

- Separate compilation UNITS are not supported.

Performance

From running the benchmarks set out in another paper, we concluded that the system performed better than a 248K byte 11/34 with a Fujitsu disk, but not as well as the same machine with an 4K byte cache.

During the 2 days, we did at one time have all 8 ports logged-in, but normally ran with 5-6 people logged in. At that level, it was noticeably slower than the 11/60 with an equivalent load, but was perfectly acceptable. The system should be fully capable of supporting 8 normal University users.

The benchmarks reveal some interesting aspects of the system's performance. In particular, creating a process takes much longer than on a PDP11. This may be due to cache turnover problems; in addition to the in-core disk cache (only 18 blocks), the disk controller maintains a cache of something like 100 blocks.

Conclusions

Broadly, our conclusions are favourable. The combination of the hardware and the kernel provides a reliable 8-user timesharing system, with wholly adequate performance. This system costs less than four average single-user systems, yet provides better performance and a vastly superior software environment. Most of the utilities functioned as expected, and the Pascal system seems to provide software compatibility with existing UCSD systems.

On the other hand, we have two major reservations about the software. Firstly, neither of the two C compilers are remotely acceptable. It is clearly impossible to move the body of applications code we have developed to this system using either compiler. We hail to understand why the Portable C compiler was not used to make the port of UNIX, or why it has not yet been made available, especially since this would also provide "f77" and "lint".

Secondly, providing the system as a single object module makes it impossible to change. This is particularly objectionable as regards the teletype driver. UNIX users in Europe, and to some extent in the US, have for many years rejected the Bell driver. In particular, the EUUG distribution of V7, which is used in the majority of European V7 sites, includes the driver upon which all our communications developments are based. We would find it difficult to accept any system which did not permit us to use this driver.

IMPORTANT ANNOUNCEMENT IMPORTANT ANNOUNCEMENT

Special Issue of AUUGN on UNIX for Computer Science Teaching

Editor : Judy Kay

We are seeking submissions in a wide range of topics that come under this subject. In particular, we plan to print reports in the following areas:

- good novice environments available under UNIX
- tools that are useful in teaching and administration in student student environments
- automatic grading systems
- experience with compilers/languages available under UNIX (eg. pi and pc Pascal, f77, euclid, Cobol)
- experience with editors available (eg. ed, ded, vi, EMACS and structure editors)
- approaches to security and control
- other reports of teaching experiences,

We welcome reports of on-going projects as well as those that have been completed. We expect that valuable contributions could be provided from non-educational sites and we trust that educational UNIX users will deluge us with brilliant articles.

We are confident that this issue will become a valuable resource for the many of us involved in Computer Science teaching.

Submissions are preferred in camera-ready form by 31st, May, 1983. They should be sent to:

Judy Kay,
Basser Department of Computer Science, F09.
University of Sydney, 2006.

or mailed to judy:basservax

and you may contact me at (02)-692-3525 or 692-3424

VMS Version 3

sultan!dag

Please stop submitting SPR's. This is our system. We designed it, we built it, and we use it more than you do. If there are some features you think might be missing, if the system isn't as effective as you think it could be, TOUGH! Give it back, we don't need you. See figure.

```
 -------------------------------
!                               !
!              _                !
!             { }               !
!             | |               !
!             | |               !
!          .-.! !.-.            !
!        .-!  ! !  !.-.         !
!        ! !       !  ;         !
!        \            ;         !
!         \          ;          !
!          !         :          !
!          !         |          !
!          |         |          !
!                               !
!_______________________________!
```
Figure 1.

Forget about your silly problem, let's take a look at some of the features of the VMS operating system.

1) Options. We've got lots of them. So many in fact, that you need two strong people to carry the documentation around. So many that it will be a cold day in hell before half of them are used. So many that you are probably not going to do your work right anyway. However, the number of options isn't all that important, because we picked some interesting values for the options and called them ...

2) Defaults. We put a lot of thought into our defaults. We like them. If we didn't, we would have made something else be the default. So keep your cotton-picking hands off our defaults. Don't touch. Consider them mandatory. "Mandatory defaults" has a nice ring to it. Change them and your system crashes, tough. See figure 1.

3) Language Processors. They work just fine. They take in source, and often produce object files as a reward for your efforts. You don't like the code? Too bad! You can even try to call operating system services from them. For any that you can't, use the assembler like we do. We spoke to the language processor developers about this, they think a lot like we do, they said "See figure 1".

4) Debuggers. We've got debuggers, one we support and one we use. You shouldn't make mistakes anyway, it is a waste of time. We don't want to hear anything about debuggers, we're not interested, See figure 1.

5) Error Logging. Ignore it. Why give yourself an ulcer? You don't want

to give us the machine to get the problem fixed and we probably can't do it anyway. Oh, and if something breaks between 17:00 and 18:00 or 9:30 and 10:30 or 11:30 and 13:30 or 14:30 and 15:30 don't waste your time calling us, we're out. See figure 1.

6) Command Language. We designed it ourselves, it's perfect. We like it so much we put our name on it, DCL - Digital's Command Language. In fact we're so happy with it, we designed it once for each of our operating systems. We even try to keep it the same from release to release, sometimes we blow it though, See figure 1.

7) Real Time Performance. We got it. Who else could have done such a good job? So the system seems sluggish with all those priority 18 processes, no problem, just make them priority 1. Anyway, realtime isn't important anymore like it used to be. We changed our groups name to get rid of the word realtime, we told all our realtime users to see figure 1 a long time ago.

In conclusion, stuff your SPR. Love VMS or leave it, but don't complain.

-adapted from TOPS-20

A 'secure program'

A note and extension to UNIX V7 printf: the %r format.

Bob Buckley

School of Maths and Physics,
Macquarie University,
North Ryde, NSW.

Introduction.

A few astute hackers may have noticed that the PDP-11 stdio library routine which implements printf, fprintf and sprintf includes code for a '%r' format. This undocumented feature is mostly unknown and unused.

Apart from the lack of documentation, in its distributed form it isn't particularly useful. This short article describes a simple improvement and demonstrates a sample usage for this feature*.

The standard '%r' format.

The original source code (probably the file doprnt.s in directory .../lib/libc/stdio) suggests the 'name' of this format is *remote*. When '%r' is encountered in a format string, the corresponding argument is assumed to be the address of a vector containing firstly, a pointer to a new format string, followed by any arguments which will be needed during the processing of the format string. The arguments are assumed to be in the same form as they would appear in an argument list passed to a procedure (ie. a char is a short, an unsigned is a long, etc. on the PDP-11).

As an example, consider the following program:

```
main( )
{
        int vec[2];
        vec[0] = (int) "a %dnd line\n";
        vec[1] = 2;
        printf("line %d\n%r", 1, vec);
}
```

When run, this program produces the following output:

```
line 1
a 2nd line
```

Of course, lists can be constructed and behave as one would hope.

* A *feature* is a bug that looks like it might be useful, eg. most computer manufacturers allow many features to remain in both the hardware and software of their computer systems. Heavy use of *features* is probably the best available protection for software - it makes it unmodifiable, unportable and unmaintainable.

Notice, that putting anything after a '%r' has no effect.

The number of uses for such a feature is limited. Perhaps the most sensible is in something like

```
error(fmt, a, b, c)
char *fmt;
{
        fprintf(stderr, "error in %s(%d):%r",
                        filename, lineno, &fmt);
}
```

This is probably the intended usage of this format. However, it is potentially more general in its application if the semantics are extended slightly.

An alternative implementation.

Consider the effect of making '%r' *recurse* rather than *remote*. When the format list and arguments corresponding to a '%r' in a format string has been completely processed, formatting continues where it left off in the previous string and argument list.

For simple cases, the behaviour will be the same as before. But more complex structures can now be constructed which allow considerable flexibility in format conversion.

The implementation of this is simple (the biggest difficulty is understanding the largely uncommented code in the original assembler routine), and the costs are few (about 3 words of stack space for each recursion level, about 10 instructions in the routine and a couple of instructions in each format conversion at run time). The efficiency conscious might want to remove tail recursion though it hardly seems to matter.

An application.

Of course, this is all pointless unless it is somehow useful. Perhaps one of the most elegant uses for this feature is as a replacement for 'bundling'*. The mechanism can be used to perform the output of complex tree and acyclic graph structures. In particular, it can simplify the writing of small compilers and translators.

Consider the following simple translator for assignment statements which directly produces code for a PDP-11.

```
%token ASSIGN ADD name
%{
char  list[] =      "%r%r",
      assign[] =    "%r%rmov (sp)+,*(sp)+\n",
      rv[] =        "%rmov *(sp)+,-(sp)\n",
      add[] =              "%r%radd (sp)+, (sp)\n",
      lvlocal[] = "mov r5,-(sp)\nadd $%d,(sp)\n",
      lvstatic[] ="mov $%s,-(sp)\n",
      rvlocal[] = "mov %d(r5), -(sp)\n",
```

* Bundling is a programming technique which compactly represents trees of strings. It was described in UNIX V6 TMG and YACC documents.

```
        rvstatic[] ="mov %s, -(sp)\n";
%}
%%
prog: stlist = { printf("%r", $1); }
    ;
stlist:     st
    | stlist st = { $$ = node(3, list, $1, $2); }
    ;
st:   lv ASSIGN exp = { $$ = node(3, assign, $1, $3); }
    ;
exp:  lv = {if($1->op == lvlocal)
                    $1->op = rvlocal;
              else if($1->op == lvstatic)
                    $1->op = rvstatic;
              else
                    $$ = node(2, rv, $1);
             }
    | exp ADD lv = { node(3, add, $1, $3); }
    ;
lv:   name = { switch($1->syntype) {
                  case LOCAL:
                    $$ = node(2, lvlocal, $1->offset);
                    break;
                  case STATIC:
                    $$ = node(2, lvstatic, $1->name);
                    break;
                  default:
                    error("bad lv");
               }
              }
    ;
```

Clearly, this doesn't produce the most elegant code but it was quick to write and is simple. It performs at as efficiently as most comparable techniques.

Similar mechanisms can be constructed for many complex data structures resulting in easy conversions to external representations.

Conclusions.

If you are looking for clever programming techniques or corner cutting tools, this one may prove useful. This technique was used to produce a small compiler for undergraduate teaching. The compiler was simple and fast, demonstrating that things need not always be difficult. Students had few difficulties in understanding the compiler, and doing simple modifications in a single term.

Of course, this mechanism can be used to produce extremely obscure code. It is a technique of little significance, but can sometimes save considerable effort. It is regrettable that a corresponding mechanism can't be built into the scanf routines. It is also a pity that it is difficult to keep portable.

Tips for writing programs that won't port to VMS and the DEC C compiler

R. Grevis

University of New South Wales

- As the DEC compiler does not have them, use bit fields fully. Just as importantly, use the actual bit field some of the time, and mask it yourself the rest of the time, so that removing the bit field packing of the declaration will cause the program to fail.

- Use upper and lower case global names of the same type. The VMS compiler and loader do not distinguish case, so havoc can be wrought by choosing appropriate naming schemes. A handy idea is renaming those temporary local variables like "temp", "i", and "f", to be the same as global variable names except in case. The compiler won't even detect this error, and they certainly don't have DEC lint.

- The VMS loader requires that space be declared once only, so the normal trick of having an include file of var declarations will produce a pleasant number of pages from the loader complaining about multiple declarations.

- Use Unix. Stat, dir, and info like that are not avalable as transliterating library procedures (DEC softpersons take note if you can), so intertwine such things in your code. They are a pain in the neck to remove.

- VMS is set up to detect more overflow conditions, so try to get integer overflow occuring whenever possible. The VMS program will explode.

- Simply use the Unix philosophy. Producing processes and pipelines easily is a foreign concept to VMS, so write programs which use other tools like grep, sort, tail, sed, etc, and also divide the problem into separate communicating processes when appropriate.

If programmers follow the simple rules above, its likely you will end up with a Unix portable piece of software which could NEVER run with DEC C and VMS without an intimate understanding of what the program does. And that should really make those system managers foam at the mouth who have just spent $10,000 on a single compiler.

What can you say?... They want C to get some of the tools, and that's like dropping a Ferrari engine into a Cadillac. It will sure improve the Cadillac, but what's needed is for them to take a step back and see what they really should be doing.

Long division in UNIX V7 on the LSI-11/23.

Bob Buckley

School of Maths and Physics,
Macquarie University,
North Ryde, NSW.

A problem and a solution.

Once upon a time (several weeks ago) I noticed that our poor little LSI-11/23 wasn't always dividing by 2 correctly. The problem occurred when dividing longs. On checking, it I discovered that a very clever piece of code was used to implement division of longs by small numbers.

Unfortunately the clever code comes unstuck as the LSI-11/23 doesn't produce the same results as its bigger siblings when overflow occurs during a DIV instruction (I had heard rumours that this was the case in Europe but hadn't experienced the problem so didn't take much notice). Further investigation revealed that the instruction 'div r3,r0' on the LSI-11/23 was changing the value of r1 if the V-bit gets set. It seemed that the library routines didn't expect this.

Clearly, the only thing to do was to consult the *Microcomputer processor handbook* to find out what the differences were. Of course, the relevant entry looks like it had been turned into pumpkin scone mixture by a computerised typesetting system; so that didn't help. The supplier of the equipment had no idea whether we had a fault or a design 'feature' as their books are just the same as ours. Next, one tries to obtain the information from the manufacturers, but this is futile, as they first ask you what operating system you run and from whom you bought your computer. As neither answer is particularly pleasing to them communication effectively stops.

Fortunately, the UNIX software is easily fixed. The division and remainder routines (these are part of the C runtime library .../lib/crt called ldiv.s, aldiv.s, lrem.s and alrem.s; and part of the machine code support for the operating system called ldiv and lrem) all deal with the low order 16 bits with a sequence of instructions which begins

```
div   r3,r0
bvc   1f
sub   r3,r0
```

The routines can be fixed by inserting the instruction

```
mov   r2,r1
```

between the 'bvc' and the 'sub' instruction.

The code is longer than the original, and a little slower, but appears to give correct results on our LSI-11/23. It is quicker and smaller than my other attempts at PDP-11 compatible versions of the routine. This code relies on r0 remaining unchanged. The manuals don't seem to guarantee this, so a comment to this effect should be added to the code. If you like using these types of features you might find it shorter in the remainder routines to remove the code which deals with overflow (rather than inserting the instruction given above) as r1 seems to have the desired value anyway. Personally, I think this is not a very good idea, as it isn't 'upwards compatible'.

What should be done?

Of course, the routines should be changed in all LSI system libraries. All programs which use division of (or by) longs (or unsigneds) should be recompiled.

Other PDP-11s should be checked to see if there are any problems with them (has anyone checked the PDP-11/44 yet?). If you are in the habit of distributing binaries, then you should fix your library even on a PDP-11. As far as I can tell, this code should work equally well on all relevant members of the PDP/LSI-11 family.

I ran into the problem when dividing a long variable by the constant 2. I was surprised that the compiler didn't optimise this to a shift. Perhaps the compilers should be changed to produce shifts and mask operations for division and remainders by constants which are powers of two in suitable cases.

We don't have the support routines for unsigned longs on our system. I suspect that these suffer the same problem. If you have a working pcc on one of these tiny machines then you may need to fix these routines as well.

Conclusions.

The conclusions are many: that assembler code is a pain in the neck (the code here was poorly documented and far from clear); toy computers usually screw up when asked to do anything that isn't completely basic; etc. More importantly, many systems run with significant bugs for a long time before they are detected. The clarity of documentation for the PDP/LSI-11 instruction set could still be improved after all these years. This sort of difference could be properly documented.

Clearly, a set of verification programs is needed for testing C language implementations and different models of hardware. This sort of error should be detected during the porting of the system, not a long time afterwards.

An opinion.

Whether it be official or not, DEC's negative attitude to UNIX sites helps no-one. This attitude exists in Europe as well as in Australia, and is far from beneficial. UNIX users who experience this won't have much 'brand loyalty' and are probably in a better position than most to change to cheaper hardware. A more professional attitude from DEC in this area produces a better service for the users of their equipment and DEC is likely to maintain its strong position as a supplier of UNIX hardware. This should benefit both parties at a very low cost.

Spontaneous Bernoulli

A Structuralist Language

Karlos Mauvtaque

Department of Ergonomics and Aesthetics

University of Chippendale at California (UCC)

Ever since primitive man's first scratchings in the dirt there has been a need for a unification of thought. Previous unstructured life-forms lacked the appreciation of how true modularity could improve the quality of life and how long and tedious its variable qualifiers could be.

But alas with the unification of primitive man's creativity came the stifling of inspiration. Life lacked that spontaneous zest that made those primitive scratchings into commercially viable software.

UNIX[1] is really no more than a little pile of gravel - a by-product of those scratchings. C[2] is a lot of carp,[3] Pascal[4] is a brand of sweets. Spontaneous Bernoulli must be the language for today, TUNAS[5] the operating system for the post-Brezhnev Western world.

The C language, a product of those keen lads at Bell[6] Laboratories, has become popular because of the blatant immorality of many of its constructs. The ability to transform an arbitrary constant into a pointer to just about anything smacks of hedonism, the kind of unrestrained perversity that was the downfall of the Roman Empire.

Pascal, a product of Nick and his cronies in collaboration with Control Dada Carporation, reeks of self imposed puritanism. Primitive man is not allowed to put his stick in the dirt just in case he gets it dirty. This sort of bondage and discipline denies the programmer the simple pleasures of initialised data, and the data types are only really suited to 60 bit sticks.

Spontaneous Bernoulli is designed to be an aware, open, communicative language - a language for the occasional user or the potential addict. No side effects are known (nor are they tolerated). Its beauty lies in structure, its structure in form, the form is art and the art is beautiful.

The non-euclidean structure of SB solves all of the problems of other

1 UNIX is a big shop with lots of shelves.

2 Delightfully unpredictable.
'Cie' - Candice Burgermeistersinger.

3 A carp is a freshwater fish (Bishop)
with a funny head.

4 Pascal is a trademark of Cadbury-Schweppes.

5 It's the fish (Bishop) that John West reject.

6 The klangorous timbre of the bell is due
to inharmonic partials distressing the cochlea.

langauges proposed to solve all the problems of other languages. For example Discurrent Driclad, a language proposed by those valiant hacks at the University of Troll-hits, makes incorrect suppositions about the habits of friendly little furry woodland animals. In fact the second term of axiom 9 should read;

$$\sum_{i=0}^{i=\infty} \frac{\alpha_i+\beta_i}{\text{Bushy Tail}}$$

After all, not all of us can see it through the winter without a supply of nuts!

Let us examine the consequences of this axiom.

DD statement	Purpose	Result
a := b	Set a to the value of b	Bind a to a function which returns the current value of b, invariant of future change of b.
if i = j	Test equality of i and j.	Compare the equivalence of the function bindings of i and j. (Each of i and j being bound as above).
stop(war);	Rid the third world of all aggression. Unite the masses, striving for peace.	Stop the program and head for the basement.

I am sure you had no trouble experiencing the disorientation of that example.

What does SB have to offer? It is designed with the creative programmer in mind. No abstraction is too frivolous for the silicon-chip hot-tub age that we live in. Self indulgence is not only supported, it is actively encouraged. A marketing expert system system (MESS) allows the programmer an infinite choice between a tight, up-market, chatty little number and a laid-back, middle-of-the-road lemon.

Semantic and syntactic restraints are derived from the programmers intent. This effectively eliminates many erors associated with traditional programming languages.

Spontaneous Benoulli, formal systems in informal terms. Try it, you'll like it.

Roster of Exhibitors

Winter 1983 UNICOM Conference

ABLE COMPUTER
1732 Reynolds Avenue
Irvine, CA 92714
Attn: Robert T. Jones
(714) 979-7030
Booth # 418

AGS COMPUTERS, INC.
1135 Spruce Drive
Mountainside, NJ 07092
Attn: Chad Henderson
(201) 654-4321
Booth # 520

ALCYON CORP.
1716 Production Ave.
ian Diego, CA 92121
Attn: Lovell C. Chase, Jr.
(619) 578-0860
Booth # 519

ALTOS COMPUTER SYSTEMS
360 Bering Drive
ian Jose, CA 95131
Attn: Glenda Stroup
(408) 946-6700
Booths # 511, 513

MDAHL CORP.
Post Office Box 470
M/S: 215
unnyvale, CA 94086
Attn: Joan Rogers
408) 746-7032
Booth # 406

VIV CORP.
Cummings Park
Voburn, MA 01801
Attn: Ed Arsenault
617) 933-1165
ooth # 217

RS
200 Route 7
atham, NY 12110
Attn: Tom Topolinski
518) 783-1161
ooth # 518

ADRE SYSTEMS, LTD.
4542 Ventura Blvd., #205
herman Oaks, CA 91403
ttn: Karl Klessig
13) 789-8588
ooth # 515

CALLAN DATA SYSTEMS
2645 Townsgate Rd.
Westlake Village, CA 91361
Attn: Kenn G. Morris
(805) 497-6837
Booths # 400, 402

CHARLES RIVER DATA SYSTEMS, INC.
4 Tech Circle
Natick, MA 01764
Attn: W. Daniel DeLea, Jr.
(617) 655-1800
Booth # 114

CODATA SYSTEMS CORP.
285 N. Wolfe Road
Sunnyvale, CA 94086
Attn: Rosemarie Calpin
(408) 735-1744
Booth # 404

COMPUTER CONSOLES, INC.
1212 Pittsford-Victor Road
Pittsford, NY 14534
Attn: Renee Wright
(716) 248-8200
Booth # 313

COMPUTER TECHNOLOGY GROUP-TELEMEDIA, INC.
310 S. Michigan Avenue
Chicago, ILL 60604
Attn: Robert E. Hinchey
(312) 987-4092
Booths # 204, 206

CORVUS SYSTEMS
2029 O'Toole Avenue
San Jose, CA 95131
Attn: C. Stenman
(408) 946-7700
Booths # 501, 503

COSMOS SYSTEMS, INC.
525 University Ave., # A60
Palo Alto, CA 94301
Attn: Diane Doran
(415) 326-9150
Booths # 505, 507

CYB SYSTEMS, INC.
6448 Hway, 290 E., #D-106
Austin, TX 78723
Attn: Vicki Stogsdill
(512) 458-3224
Booths # 317, 319

DIGITAL EQUIPMENT CORP.
129 Parker Street (PK03-1/M36)
Maynard, MA 01754
Attn: Jack Richardson
(617) 493-8207
Booths # 310, 312, 314, 316

DUAL SYSTEMS CORP.
2530 San Pablo Avenue
Berkeley, CA 94702
Attn: Elizabeth Sumner
(415) 549-3854
Booth # 202

FORTUNE SYSTEMS CORP.
300 Harbor Blvd.
Belmont, CA 94002
Attn: Dan Elliston
(415) 595-8444
Booth # 302

GOULD S.E.L.
COMPUTER SYSTEMS DIVISION
6901 W. Sunrise Blvd.
Ft. Lauderdale, FL 33310
Attn: Teri-Lisa Maidhof
(305) 473-1050
Booths #113,115,117,119,121

HEWLETT-PACKARD COMPANY
11000 Wolfe Road
Cupertino, CA 95014
Attn: Neal Kuhn
(408) 257-7000
Booth # 407

HUMAN COMPUTING RESOURCES CORP.
10 St. Mary St., #401
Toronto, Ontario
M4Y 1P9 CANADA
Attn: Edward Borkovsky
(416) 922-1937
Booths # 101, 103

IBM CORP.
23rd Floor
425 Market Street
San Francisco, CA 94105
Attn: Bill Clinton
(415) 545-4634
Booth # 500A

IMAGE NETWORK
1633 Bayshore Hwy, Suite 239
Burlingame, CA 94010
Attn: John Copeland
(415) 665-6426
Booth # 600

INSTITUTE FOR ADVANCE PROFESSIONAL STUDIES
55 Wheeler Street
Cambridge, MA 02138
Attn: Dr.Donald D. French
(617) 497-2075
Booth # 415

INSTRUMENTATION LABORATORY INC., PIXEL DIVISION
113 Hartwell Ave.
Lexington, MA 02173
Attn: Laurel Dutcher
(617) 861-0710
Booth # 304, 306

INTEL CORP.
3065 Bowers Ave.
Mail Stop SC 4-718
Santa Clara, CA 95051
Attn: Lucille Harris
(408) 496-7162
Booth # 311

INTERACTIVE SYSTEMS CO
1212 Seventh Street
Santa Monica, CA 90401
Attn: Dwayne C. Lowry
(213) 450-8363
Booths # 508, 510

INTERLAN INC.
3 Lyberty Way
Westford, MA 01866
Attn: Gerald W. Wesel
(617) 692-3900
Booth # 112

INTERNATIONAL DATA SERVICES, INC.
1020 Stewart Drive
Sunnyvale, CA 94086
Attn: Richard J. Cavanaugh
(408) 730-UNIX
Booth # 500

LOGICAL SOFTWARE, INC.
55 Wheeler Street
Cambridge, MA 02138
Attn: Douglas I. Kalish
(617) 864-0137
Booths # 411, 413

MARK WILLIAMS COMPANY
1430 W. Wrightwood
Chicago, IL 60614
Attn: Karen Gunn
(312) 472-6659
Booth # 219

MASSCOMP
543 Great Road
Littleton, MA 01460
Attn: Cathy Pfister
(617) 486-9425
Booths # 301, 303

METHEUS CORP.
5289 N.E. Elam Young Pkway
Bldg. D-600
Box 1049
Hillsboro, OR 97123
Attn: Paul Chen
(503) 640-8000
Booth # 604

MICROSOFT CORP.
10700 Northup Way
Bellevue, WA 98004
Attn: Patricia McGinnis
(206) 828-8080
Booths # 401, 403

MOMENTUM COMPUTER
SYSTEMS INTERNATIONAL
965 W. Maude Avenue
Sunnyvale, CA 94086
Attn: Ron Grondona
(408) 245-4033
Booths # 105, 107

NATIONAL
SEMICONDUCTOR CORP.
2900 Semi Conductor Drive
M/S 16250
Santa Clara, CA 95051
Attn: William Brennan
(408) 721-5752
Booth # 517

NETWORK RESEARCH CORP.
1964 Westwood Blvd., #200
Los Angeles, CA 90025
Attn: Howard Gordon
(213) 474-7717
Booths # 502, 504

NORTH AMERICAN
TECHNOLOGY, INC.
9570 S.W. Barbur
Portland, OR 95825
Attn: Roberta Donovan
(503) 245-6585
Booth # 315

ONYX SYSTEMS, INC.
25 E. Trimble Road
San Jose, CA 95131
Attn: Mike Spies
(408) 946-6330
Booths #100,102,201,203

PACIFIC
MICROCOMPUTERS, INC.
119 Aberdeen Dr., #7
Cardiff, CA 92007
Attn: John Metzger
(619) 436-8649
Booths # 305, 307

PARALLEL COMPUTERS
501 Cedar Street
Santa Cruz, CA 95060
Attn: Scott D. Pine
(408) 429-1338
Booths # 601, 602

PERKIN-ELMER
2 Crescent Place
Oceanport, NJ 07757
Attn: Emily Backus
(201) 870-4717
Booths # 106, 207

PLEXUS COMPUTERS, INC.
2230 Martin Avenue
Santa Clara, CA 95050
Attn: Leslie D. Schroeder
(408) 988-1755
Booths # 417, 419

REAL TIME SYSTEMS
Elliott Terrace Workshops
Newcastle Upon Tyne
ENGLAND NE4 6UP
Attn: Bill Colwell
0632-733131
Booth # 318

RELATIONAL DATABASE
SYSTEMS, INC.
1208 Apollo Way # 503
Sunnyvale, CA 94086
Attn: O. D. Parkinson
(408) 746-0982
Booth # 300

RELATIONAL
TECHNOLOGY, INC.
2855 Telegraph Ave., # 312
Berkeley, CA 94705
Attn: Debbie Muzio
(415) 845-1700
Booth # 606

RYAN-McFARLAND CORP.
9057 Soquel Drive
Aptos, CA 95003
Attn: Barbara Fitzgerald
(408) 662-2522
Booths # 216, 218

THE SANTA CRUZ
OPERATION INC.
500 Chestnut Street
Santa Cruz, CA 95060
Attn: John Rigdon
(408) 425-7222
Booths # 109, 111

SOFTEST, INC.
555 Goffle Road
Ridgewood, NJ 07450
Attn: David Schneider
(201) 447-3901
Booth # 110

SOFTRON, an HHB COMPANY
116 Route 17 North
Upper Saddle River, NJ 07458
Attn: Suzanne Hartig
(201) 327-8014
Booth # 116

SOFTWARE IRELAND, LTD.
31 Anderson Way
Menlo Park, CA 94025
Attn: Gordon Bell
(415) 854-2383
Booth # 214

SUN MICROSYSTEMS INC.
2310 Walsh Avenue
Santa Clara, CA 95051
Attn: Ajay Puri
(408) 748-9900
Booth # 210, 212

SYSTEM INDUSTRIES
4440 Von Karman Ave., #200
Newport Beach, CA 92660
Attn: Scot Leisy
(714) 851-6289
Booths # 410, 412

TELETYPE CORP.
47 Stonehenge
Morristown, NJ 07960
Attn: Fred P. Eick
(201) 538-6890
Booth # 603

THIRD EYE SOFTWARE
734 Webster Street
Palo Alto, CA 94301
Attn: Peter Rowell
(415) 321-0967
Booth # 405

3 COM CORP.
1390 Shorebird Way
Mountain View, CA 9
Attn: Mike Halaburka
(415) 961-9602
Booth # 514

UNIQ COMPUTER C
28 S. Water Street
Batavia, IL 60510
Attn: Roger A. Knuth
(312) 879-6515
Booth # 516

UNISOFT SYSTEMS
2405 4th Street
Berkeley, CA 94710
Attn: Sharon Hill
(415) 644-1230
Booth # 200

USER TRAINING CO
21565 Alma Court
Los Gatos, CA 95030
Attn: George Champag
(408) 354-6433
Booth # 118

VENTURCOM, INC.
139 Main Street
Cambridge, MA 02142
Attn: Paul Kleppner
(617) 661-1230
Booth # 416

VIRTUAL
MICROSYSTEMS, INC
2150 Shattuck Ave.
Suite 720
Berkeley, CA 94704
Attn: Ross Charney
(415) 841-9594
Booth # 605

WICAT SYSTEMS, IN
1875 S. State
Orem, UT 84057
Attn: Ken Sorber
(801) 224-6400
Booth # 414

YATES VENTURES
4962 El Camino Real #
Los Altos, CA 94022
Attn: Jeffrey Brown
(415) 964-0130
Booth 211

ZILOG, INC.
1315 Dell Avenue
Campbell, CA 95008
Attn: Carolyn Hayes
(408) 370-8000
Booths # 213, 215

ADVANCE MEETING ANNOUNCEMENT AND

CALL FOR PAPERS

13TH EUUG MEETING

11TH - 13TH APRIL 1983

Wissenschaft Centrum, Old Town of Badgodesburg, BONN.

e 13th EUUG Meeting will take place in Bonn on the three days of 11th, 12th, d 13th April under the organisation of Hans Grimm, GMD (Gesellschaft fur thematik und Datenverarbeitung).

e meeting will include special sessions on:-

Unix System V by AT&T Managerial and Technical Staff,
Virtual memory Unix (VAX et al),
Personal Workstations (68000 systems et al),
Development of the Unix filestore,
Secure UNIX systems,
European Network.

rited speakers include the following: -

eve Bourne, Bell Laboratories,
Birman, ex Berkeley Cocanet, now Cornell,
ll Joy, SUN Microsystems,
n Leffler, of Berkeley, who is taking over from Bill Joy,
dy Tanenbaum (held over from Leeds due to popular demand).

pers are invited from members on any of the topics mentioned above as ll as:

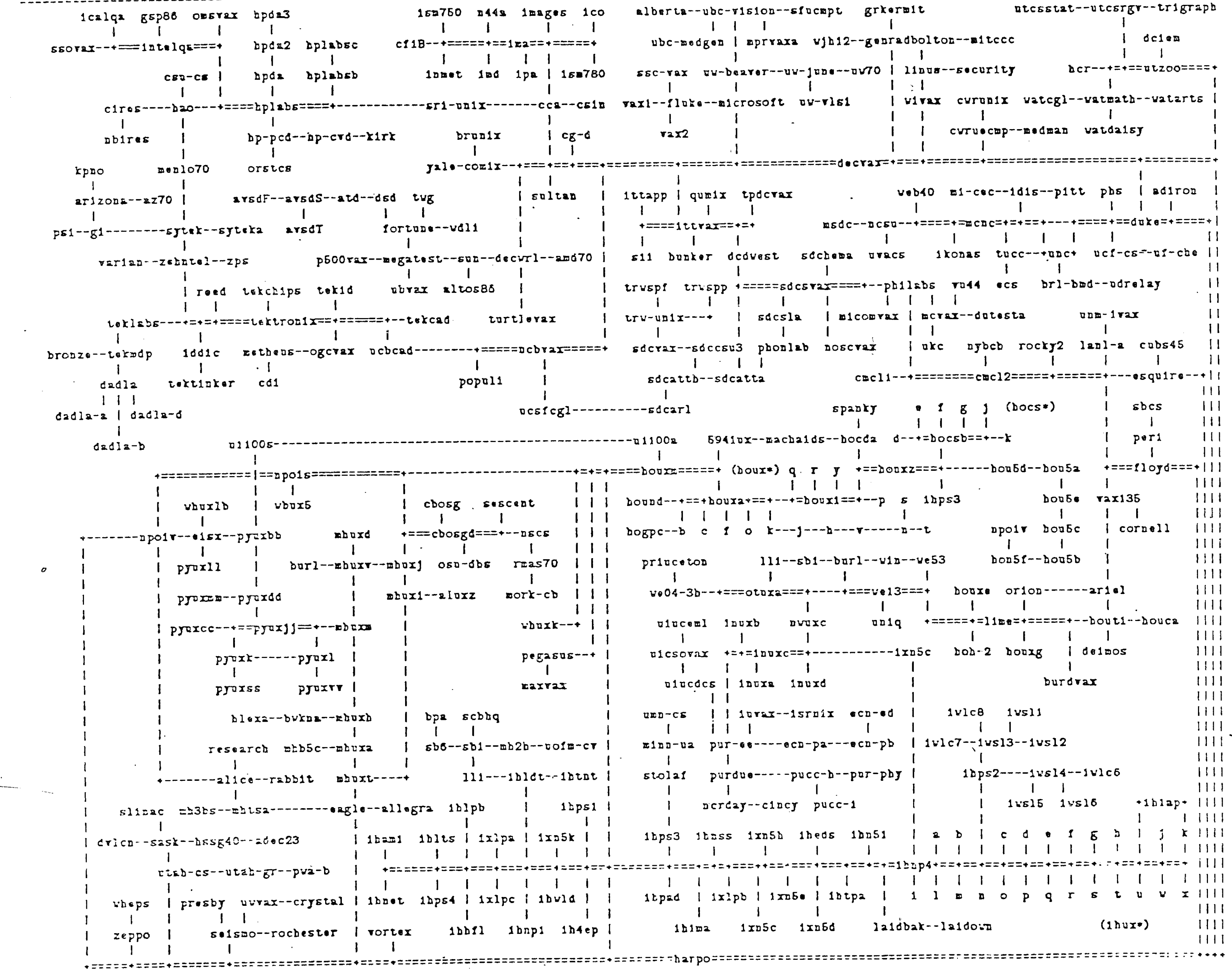
research on UNIX based projects
echnical descriptions of new UNIX software products, especially LAN and latabase products,

on any topic likely to be of interest to the membership as a whole.

pers (or initial abstracts) (or simply offers to speak) should be dressed to:

Hans Grimm
GMD
Postfach 1240
5205 San Augustin
Badgodesburg
BONN West Germany

USENET LOGICAL MAP - January 24, 1983 Karen Shannon (sun!kas)


icalqa gsp86 omsvax hpda3
ssovax intelqa hpda2 hplabsc
csu-cs hpda hplabsb
cires hao hplabs sri-unix cca csin
nbires hp-pcd hp-cvd kirk brunix cg-d
kpno menlo70 orstcs yale-comix
arizona az70 avsdF avsdS atd dsd twg sultan
psi gi sytek syteka avsdT fortune wdl1
varian zehntel zps p500vax megatest sun decwrl amd70
reed tekchips tekid ubvax altos86 turtlevax
teklabs tektronix tekcad
bronze tekmdp iddic metheus ogcvax ucbcad ucbvax
dadla tektinker cdi populi
dadla-a dadla-d ucsfcgl sdcarl
dadla-b ui100s ui100a
ism780 n44a images ico
cfi8 ima
inmet imd ipa ism780
alberta ubc-vision sfucmpt grkermit
ubc-medgen mprvaxa wjh12 genradbolton mitccc
ssc-vax uw-beaver uw-june uw70 linus security
vax1 fluke microsoft uw-vlsi vivax cwrunix watcgl watmath watarts
vax2 cwruecmp medman watdaisy
decvax
utcsstat utcsrgv trigraph
dciem
hcr utzoo
ittapp qumix tpdcvax
ittvax
sii bunker dcdwest sdchema uvacs
trwspf trwspp sdcsvax philabs
trw-unix sdcsla micomvax
sdcvax sdccsu3 phonlab ncsvax
sdcattb sdcatta
web40 mi-cec idis pitt phs adiron
msdc ncsu mcnc duke
ikonas tucc unc ucf-cs uf-che
vn44 ecs brl-bmd ndrelay
mcvax dutesta unm-ivax
ukc nybcb rocky2 lanl-a cubs45
cmcl1 cmcl2 esquire
spanky e f g j (bocs*) sbcs
5941ux machaids bocda d bocsb k peri
npois bourz (bour*) q r y houxz hou5d hou5a floyd
whux1b whux5 cbosg sescent
npoiv eisx pyuxbb mhuxd cbosgd nscs
pyuxll burl mhuxv mhuxj osu-dbs rmas70
pyuxmm pyuxdd mhuxi aluxz mork-cb
pyuxcc pyuxjj mhuxm whuxk
pyuxk pyuxl pegasus
pyuxss pyuxvv marvax
blexa bvkna mhuxh bpa scbhq
research mhb5c mhuxa sb6 sb1 mh2b uofm-cv
alice rabbit mhuxt lll ihldt ihtnt
slimac mh3bs mhtsa eagle allegra ihlpb ihps1
dvlcn sask hssg40 adec23 ihami ihlts ixlpa ixn5k
utah-cs utah-gr pwa-b
wheps presby uwvax crystal ihnet ihps4 ixlpc ihwld
zeppo seismo rochester vortex ihhfl ihnp1 ih4ep
hound houxa houxi p s ihps3 hou5e vax135
hogpc b c f o k j h v n t npoiv hou5c cornell
princeton lll sbi burl win we53 hou5f hou5b
we04-3b otuxa we13 houxe orion ariel
uiucdcs inuxb nwuxc uniq lime houti houca
uiucsovax inuxc ixn5c boh-2 houxg deimos
uiucdcs inuxa inuxd burdvax
umn-cs iuvax isrnix ecn-ed ivlc8 iwsl1
minn-ua pur-ee ecn-pa ecn-pb ivlc7 iwsl3 iwsl2
stolaf purdue pucc-b pur-phy ihps2 iwsl4 ivlc6
ncrday cincy pucc-i iwsl5 iwsl6 ihlap
ihps3 ihuss ixn5h iheds ihn51 a b c d e f g h j k
ihnp4
ihpad ixlpb ixn5e ihtpa l m n o p q r s t u w x
ihima ixn5c ixn5d laidbak laidown (ihux*)
harpo

The following info. was received in response to a request for site details from hosts on the SUN.

I have included phone numbers where these are known, so please feel free to call the respective sites for further details.

food
  Food Technology, UNSW
  LSI 11/23
  Pertec D4000 20Mb
  AED floppy controller
   (8" dbl sided dbl density)
  Sanders Media 12/7
  HP7450A (A4 plotter)
  UNIX level 6 AUSAM

bio23 (662 2668)
  Biological Sciences, UNSW
  PDP 11/23 + FPU.
  RL02
  Tektronix 4662 plotter
  UNIX level 7 AUSAM

agsm (662 0273)
  AGSM, UNSW
  VAX-780
  TU78, RP07
  CDC 9766 + Emulex SC21V
  Emulex CS-11
  Centronics 6600 printer
  UNIX 4.1BSD

comm34 (662 3440)
  Faculty of Commerce, UNSW
  DEC PDP 11/34
  CDC 80Mb, AMPEX 80Mb
  Pertec Tape drive
  UNIX level 6 AUSAM

comm40 (662 3680)
  Faculty of Commerce, UNSW
  DEC PDP 11/40
  RK05
  Pertec (20 megabytes)
  UNIX level 6 AUSAM

physiol (692 2695)
  Physiology, Sydney Uni.
  PDP 11/23
  Q bus + qniverter
  RK05, Pertec dual RK05
  DEC dual cassette
  DEC paper tape
  DEC LPS lab. peripheral system
  Tektronix 4662 plotter
  Sanders Media 12/7 printer
  Talos digitizing tablet
  UNIX level 7 AUSAM

civil (662 3034)
  Civil Engineering, UNSW
  PDP 11/40
  PERTEC disks
  UNIX level 6 AUSAM

unswpower (662 2797)
  Elec. Eng., UNSW
  PDP 11/40
  RK05, RL02, DR11b
  AR11, TA11
  UNIX level 7 AUSAM

csu40 (662 3590)
  Comp. Serv. Unit, UNSW
  PDP-11/40
  RK05J, RK05F, RX01
  TU10
  XY11 plotter
  LV11 printer/plotter
  Centronix 6600 printer
  PC11, DP11, DR11-B
  Ampex DM980 + AED 8000
  Unix level 6 AUSAM

psych44 (692 3024)
  Psychology, Sydney Uni.
  PDP 11/44 with fpu.
  CDC 9762 + EMULEX SC21
  PERTEC t9640 + EMULEX TC11
  GT40, AR-11
  NDK-4000
  UNIX level 7 AUSAM

elecvax (662 3781)
  Elec. Eng., UNSW
  VAX 11/780
  RP06, TU77
  Data Products B900 printer
  Datasystems DLP-11
  tektronix 4015-1
  UNIX 32v/4.1bsd mixture + AUSAM

cadvax (662 3781)
  Elec. Eng., UNSW
  VAX 11/780
  CDC 9766 via EMULEX SC21
  TU45, TS11, LPA11-K
  AED colour graphics terminal
  Ramtek monitor
  HP 7580a plotter
  HP 7221c flat bed plotter
  UNIX 32v/4.1bsd mixture + AUSAM

elec70a (662 3781)
  Elec. Eng., UNSW
  PDP 11/70
  CDC 9766 + EMULEX sc70
  TU16, MDB DZ, LP05
  qume micro 5
  UNIX level 7 AUSAM

elec70b (662 3781)
  Elec. Eng., UNSW
  PDP 11/70
  RP04, TE16
  UNIX level 7 AUSAM

dsl (662 3781)
  Elec. Eng., UNSW
  PDP 11/34A
  AMPEX DM9100 + MSC1100
  MDB DZ
  hp 2631a serial printer
  UNIX level 7 AUSAM

srl (662 3781)
  Elec. Eng., UNSW
  PDP 11/34A
  RL01
  UNIX level 7 AUSAM

elec35 (662 3781)
  Elec. Eng., UNSW
  PDP 11/35
  AMPEX DM9100 + MSC1100
  UNIX level 7 AUSAM

elec40 (662 3781)
  Elec. Eng., UNSW
  PDP 11/40
  RK05
  UNIX level 7 AUSAM

melb (341 5225)
  Melbourne Uni. Comp. Sci.
  VAX 11/780
  RM03, TE16
  CDC 9766 + Emulex SC21V
  UNIX Melbourne Modified 4.1a bsd

  PE3240
  CDC 'MSM 80' disk
  Ampex 40Mb
  selch, pasla
  2-line commux
  8-line commux
  800 bpi tape drive
  Data Set Adapter
  Line Printer (TALLY)
  A/D D/A converter
  Interprocess switch
  UNIX version 7 Berkelized

mathvax (662 2067)
  Mathematics, UNSW
  VAX 11/750
  RM80, RM03, TS11
  PERTEC T9640
  UNIX 4.1BSD + AUSAM

uowcsa (282 463)
  Dept. of CS, Wollongong
  PE 3230
  MSM300 (CDC 9766) 300 MB dual po
  MSM80 80 MB dual ported discs
  9trk 800bpi 45ips
  Tektronix 4006
  servogor 281 flat-bed plotter.
  4 line sync link to UNIVAC 1100/
  UNIX level 7 (a la wollongong)

uowcsb (282 463)
  Dept. of CS, Wollongong
  PE 3230
  MSM300 300 MB dual ported disc
  Pertec 10MB
  UNIX level 7 (a la wollongong)

  Logica Cambridge ring

Prince Henry Hospital (661 0111)
  11/44
  RL02
  Winchester 67Mb + Emulex SC21
  Cipher Streamer Tape + Emulex
  UNIX level 7

mhd (692 2104)
  Elec. Eng., Sydney Uni.
  11/34A
  RL01
  Priam 6650 Winchester + Emulex
  Cipher 75 ips 1600/800 bpi
  AED6200 dual density-single side
  Matrox 512x512 graphical display
  NDK-4000 printer
  UNIX level 7 AUSAM

```
ucc (692 3491)
  UCC, Sydney Uni.
  11/44
  AMPEX DM980 + EMULEX SC21
  TE10
  Printrex P600 600 lpm printer/pl
  Data Recognition optical marksen
  UNIX level 7 AUSAM

uccgraphics (692 3491)
  UCC, Sydney Uni.
  11/34
  Florida Data 600cps printer
  Versatec 2160A 18" printer/plott
  DIGIDATA 800/1600 bpi
  CDC 9762 + EMULEX SC21
  Evans & Sutherland Multi-picture
  UNIX level 7 AUSAM / RSX11-M

uccps (692 3491)
  UCC, Sydney Uni.
  11/24
  UNIX level 7 AUSAM

uccmc (692 3491)
  UCC, Sydney Uni.
  VAX11/780
  CDC 9766
  Systems Industries 9800 disc con
  Systems Industries SBI interface
  STC 800/1600/6250 bpi tape
  SI 9700 unibus tape controller
  UNIX (coming) / VMS

basservax (692 2824)
  Comp. Sci., Sydney Uni.
  VAX 11/780
  RM03, TE16, KMC11
  CDC 9766 via EMULEX SC21
  NDK 4000 printer
  TTY40 printer
  UNIX 32V + 4.1BSD + AUSAM

basser40 (692 2824)
  Comp. Sci., Sydney Uni.
  PDP 11/40
  TS03, PERTEC 5Mb
  AMPEX DM980 + AED 8000 controlle
  ABLE DMAX 16
  UNIX level 7 AUSAM

graphics (692 2824)
  Comp. Sci., Sydney Uni.
  PDP 11/34
  PERTEC 20Mb, ABLE DZ16
  UNIX level 7 AUSAM

chemeng (692 3832)
  Chem. Eng., Sydney Uni.
  PDP 11/60
  CDC RM03 look alikes
  Pertec 20MB
  TU10, RX02
  PP11 (reader only!)
  TALLY 2000
  Sanders Media 12/7
  UNIX level 7 AUSAM
```

THE AUSTRALIAN Tuesday December 7 1982 — COMPUTERS

Unix emerges as new standard language

From NICHOLAS ROTHWELL in New York

THE Bell Laboratories' Unix language for managing complex computer operations is rapidly emerging as the standard of the American computer industry.

This was helped last week by Hewlett-Packard deciding to offer the language as standard on its new 9000 line, and the launch of several Unix implementing chip sets.

And IBM has chosen its Series 1 minicomputer series as the vehicle to launch a new operating system, Cpix, derived from Unix, and is expected to move increasingly towards the Bell-developed system.

From megatest!dre Sat Feb 12 19:34:48 1983
Subject: Benchmarks of machines at Unicom
Newsgroups: net.micro

At UNICOM Yin Shih and I took the opportunity to run a simple benchmark on most of the machines on display at the vendor exhibit. This is an attempt at organizing the results.

In any benchmark it is important to know 1) what the benchmark tests and 2) whether you care about what the benchmark tests. An example of what I would consider to be a poor benchmark for most purposes is the Whetstone. The Whetstone bashes away at transcendental functions, exercising floating point load, add, and multiply instructions. Unfortunately it uses these instructions with a frequency seldom encountered even in "heavy floating point grinders" such as SPICE. So, to make a long story short, we're going to tell you what we did and what numbers we got, but we're not going to make any claims about which machine is the "fastest" or "best buy." We aren't going to try to interpret the results. Just the facts, ma'am. Also, we did this on our own and not as representatives of Megatest--so flame at us but don't get mad at the company (the only company ENTIRELY composed of Dead Heads! Long Live net.gdead!).

Now that we've covered our a**es, let's get to the fun part.

The benchmark (for better or worse) is:

```
                main()
                {
                        long i;
                        for (i=0;i<1000000;i++);
                        printf("Done.\n");
                }
```

This is obviously completely cpu bound and tests 32 bit integer arithmetic (or at least increment and compare).

We ran two tests. These were:

```
                        /bin/time cc tst.c
and
                        /bin/time a.out
```

The compile was quite i/o bound on most systems. Some machines which did very well in the loop had terrible disk access times. Others had reasonable i/o but slow processors.

The system configurations and prices stated below are, in most cases, as tested. In some cases the prices quoted us by the salesmen were for other configurations and that is given instead. In a few cases the information was unavailable. These figures may be wrong, but we tried to be as accurate as we could be. "Total" is the sum of user and supervisor time for both tests. Significant digits are what "time" gave us except for "real" times where I seem to have tacked a ".0" on the end of some numbers. The numbers were corrected for the Xenix ports on the TRS model 16, Apple LISA, and Parallel 68000 systems after we had a discussion with some Microsoft folks. It seems that the TRS model 16 has a 30 Hz clock and "time" gave numbers that were off by a factor of two for these machines because all three of them are object file compatible and they just transported utilities over without recompilation. The Microsoft people were quite knowledgeable and very helpful. Thanks, guys.

2) Some systems were running multiuser, so the "real" numbers shouldn't be used for comparisons.

The numbers, in increasing "total" order:

System & OS	Price	Config	cc real	cc user	cc supv	a.out real	a.out user	a.out supv	Total
Amdahl 470/V8 UTS and VM	na	na	5.49	0.26	0.18	1.29	0.24	0.01	0.69
SEL 32/87 32V	~$250K	na	10.0	1.0	2.5	1.0	0.8	0.0	4.3
VAX 11/780 w/o FP accel. 4.1BSD (courtesy of Rick Kiessig at Fortune Systems)	na	Emulex SC21 disk controller w/ CDC 9766s 1-2MB memory (Rick wasn't sure)	28.0	0.6	1.8	13.0	3.9	0.2	6.5
Masscomp SIII + Berkeley extensions	$28000	dual 10 MHz 68000s (for VM) with a 4KB cache 27 MB winch 1 MB floppy 512KB memory ascii terminal	5.0	1.2	2.1	8.0	7.5	0.1	10.9
VAX 11/750 w/o FP accel 4.1aBSD	~$100K	SC21 w/4 160MB Fujitsu winch 2 MB mem Emulex DH (the system I'm using now)	10.0	0.9	2.4	9.0	8.0	0.3	11.6
Perkin-Elmer 3210 V7+Berkeley extensions SCCS	~$50K	64 MB winchester 512KB memory	6.0	1.0	2.8	9.0	8.7	0.2	12.7
Parallel 68000 Xenix	$14K	10MHz 68000 8 serial ports Multibus 16MB winch	20.0	2.2	3.4	9.0	7.6	0.0	13.2

		1 MB floppy 256KB memory							
HP-9000 HP-UX (SIII)	$88000	10MB winch 64MB winch 2 MB memory 512KB floppy plotter color graphics	26.8	2.4	6.8	4.4	4.2	0.1	13.
Sun 4.1cBSD	$29,500	Sun Terminal 1 MB memory 84MB winch DCD tape drive 10 MHz 68000 mouse and upgrade to 68010 (The salesman said this would cost $2.5K)	14.6	1.4	4.5	9.4	7.8	0.6	14.
Charles River UNOS	$19,400	12.5MHz 68000 4KB cache 1MB memory 4 serial ports 10 MB winch 1MB floppy Versabus	26.7	6.1	3.4	6.3	5.7	0.2	15.
Alcyon APX Regulus	$29000	512KB memory winchester (size n/a) 68000 (speed n/a) streamer tape drive Q bus	17.0	2.7	1.0	14.0	13.1	0.0	16.
Codata 3300 V7	$14000	68000 (speed n/a) 2 serial ports 12 MB winch 1 MB floppy 320K memory	20.0	3.6	4.5	11.0	10.5	0.1	18.
Pacific Micro Super Sun Unisoft V7	$13,900	68000 (speed n/a) 20 MB winch 1MB floppy Multibus 256KB memory	30.0	3.5	4.6	11.0	10.5	0.1	18.
Onyx C8002 V7+Berkeley extensions	$17,000	4 MHz Z8000 20 MB winch DCD tape drive 256KB memory	13.1	2.4	3.9	13.5	13.2	0.1	19.

	VAX 11/730 4.1BSD	na	2-RL02 drives memory size n/a	14.0	2.1	4.1	33.0	14.3	1.0	21.5
13.	lcyon APS egulus	$9950	512KB memory 3 serial ports 10MB removable winchester 8 MHz 68000+ 68451 MMU 1 wait state	19.0	10.1	0.3	12.0	11.4	0.0	21.8
14.	ilog S8000 odel 11 eus (V7 + erkeley xtensions)	$17000	Z8001 (speed n/a) 18MB winchester DCD tape ZBI bus 256K memory	28.0	2.2	4.7	16.0	14.9	0.1	21.9
2 15.	ltos 586 enix	$7990	10 MHz 8086 512K memory minifloppy 10 MB winch 5-8 users	18.0	1.0	4.3	17.0	16.7	0.2	22.2
0 16.	icat 68000 nisoft 7+SIII+Berkeley	$17000	8 MHz 68000 up to 3 users 1 MB memory 1 MB floppy 15MB winch	26.0	2.8	5.8	15.0	14.7	0.1	23.4
	ltos 68000 enix	$12990	8 MHz 68000 20 MB winch 512KB floppy 16 serial ports 512K memory	24.9	5.5	8.8	10.2	9.8	0.1	24.2
1 18.	ixel 7+Berkeley xtensions	~15-18K	8 MHz 68000 1 wait state 512K memory 20 MB winch 2-1MB floppies 4 serial ports	35.0	4.9	6.4	21.0	12.6	0.3	24.2
.1 18	lexus P/25 III+PWB	$16900	Z8000 (speed n/a) 512KB memory 32MB winch DCD tape	24.6	3.7	3.4	17.2	17.1	0.1	24.3
.1 19.	ortune 32:16 7 (unbundled)+ xtensions	~$10000	5.5 MHz 68000 5MB winch 800KB floppy	19.0	3.2	7.9	15.0	14.0	0.1	25.2
	nyx C8002A	$15490	6 MHz Z8000	16.8	2.8	5.9	16.9	16.6	0.1	25.4

SIII		(speed n/a) 256K memory 5 serial ports 20 MB winch DCD tape drive							
TRS Model 16 Xenix	~$10000	68000 (speed n/a) up to 3 users 12 MB winch 1 MB floppy 256K memory	24.0	3.2	6.0	17.0	16.1	0.3	2
Apple Lisa Xenix	~$10000	5 MHz 68000 1 MB memory 5 MB winch 1 MB floppy	60.0	6.1	7.6	16.0	12.1	0.1	2
Apple Lisa Unisoft	~$10000+ cost of second drive (n/a)	5 MHz 68000 1 MB memory 2-5 MB winch 1 MB floppy	39.0	5.3	7.2	15.0	14.5	0.1	2
Dual Unisoft (reconfiguration rights cost $2000)	$16660	8 MHz 68000 (10MHz planned) S100 bus 20 MB winch 1 MB floppy 512K memory 4 users	21.0	5.9	6.1	18.0	17.2	0.2	2
BBN C/60 BBN Unix	$53000	8 users 68MB disk 512KB memory The salesman said this machine was running with C/70 microcode.	9.0	1.4	1.9	27.0	26.0	0.1	29
Momentum Hawk model 32/4 V7+Berkeley extensions+f77	$18000	6 MHz 68000 (8 MHz planned) 1 MB memory 20 MB winch 4 terminals	33.0	4.2	12.6	16.0	12.2	1.6	30
IBM PC+Sritek 68000 card Xenix	~$8000	8 MHz 68000 1 wait state Davong 5 MB winch 2 floppies 512KB memory	38.0	8.9	14.8	11.0	10.6	0.2	34
Intel 86/330X Xenix	$16000	8 MHz 8086 Multibus 2 users	16.0	3.0	4.8	37.0	28.6	0.2	36

		384 KB memory 35 MB winch							
Cosmos Lyra V7 (SIII planned)	$16500	68000 (speed n/a) 20 MB winch 1 MB floppy Multibus 512K memory 8 serial ports	34.0	7.9	8.2	23.0	21.7	0.2	38.0
IBM PC Mark Williams' Coherent	~$6000	5 MHz 8088 512K memory Corvus 10 MB winch IBM monochrome crt	26.0 (est.)			26.8	26.0	0.5	39.5 (est.
National NS16000 4.1BSD	n/a	6 MHz NS16032 1.5MB memory 20 MB winch DCD tape drive joystick and graphics display	37.0	4.4	15.5	21.1	19.6	0.6	40.1
PDP 11/23Plus DEC V7M-11 (~$1K per user license fee)	n/a	11/23Plus cpu 2-RL02s size of memory n/a	26.0	3.3	8.3	30.0	29.2	0.2	41.0
Corvus Unixplex Unisoft	~$13-14K OEMs only	8 MHz 68000 512KB memory 2 serial ports 10 MB winch	40.3	3.5	35.0	14.2	12.5	1.6	52.6

We would appreciate any additional information or corrections to this list.

Dave Emberson
Megatest Corp.
477 Potrero Rd.
Sunnyvale, CA 94086
ucbvax!lbl-csam!megatest!dre

Yin Shih
Megatest Corp.
477 Potrero Rd.
Sunnyvale, CA 94086
ucbvax!lbl-csam!megatest!yin

From sdchema!jmcg (Jim McGinness) Thu Jan 13 20:34:41 1983
Subject: UNICOM Keynote Speech--Abstract and Bibliography
Newsgroups: net.usenix

Software Army on the March - Project Strategies and Tactics

John R Mashey
Bell Laboratories
Whippany, NJ 07981

This talk describes the work of an army building roads ("software projects") through a part of the countryside that:

* seldom has current maps.

* is plagued by earthquakes ("major environment changes"), flash floods ("temporary problems"), and fog ("inability to predict the future").

* contains villages of natives ("users") who may greet roadbuilders with anything from great interest to outright hostility, but whose cooperation is essential.

* may be overrun by enemy forces ("competitors") trying to build their own roads to the same locations.

An effective campaign has two aspects:

* Doing the right thing, i.e., fighting the right war in the right place, and choosing good routes to reach the goals.

* Doing it right, i.e., building maintainable roads with adequate capacity, at a reasonable cost, without losing too many casualties, and without offending the natives.

Most software methodologies emphasize the second part; this talk emphasizes the first by examining decision processes and methods of analyzing routes. Two different veiwpoints are used. The first is the formal game theory viewpoint--making decisions in a nondeterministic, multi-stage, N-person, non-zero-sum game played with incomplete information. The second is the "army" model described above. From this viewpoint will be discussed such issues as:

* The need for scouts on motorcycles ("fast prototypers").

* How campaigns differ, and thus affect choice of troops, ranging from commando raids through the march of the hordes.

* Special precautions for earthquake territory.

* Getting natives to buy and drive your trucks, instead of shooting your tires out as you drive through their villages.

There exist many similarities in the decision processes of formal games analysis, military planning, and project management. This talk uses the first and second to help shed light on the third. Provided with the talk is an annotated bibliography, which includes a famous treatise on "project management" written in 500 B.C.

Bibliography

BLA78A Blake, S. P., Managing for Responsive Research and Development, W. H. Freeman, San Francisco, 1978.
Chapter 4 (Strategies, risk analysis) especially good: alpha strategy (careful frontend analysis) vs beta (prototyping, adaptation to state-of-the-art changes). Example of Sidewinder development (simple, reliable, cost-effective U.S. air-to-air missle) good example of beta strategy.

BUS65A Busacker, R. G., Saaty, T. L., Finite Graphs and Networks: An Introduction With Applications, McGraw-Hill, New York, 1965.
See applications to games and puzzles in sections 6-8 to 6-13.

CLA51A Clarke, A. C., Superiority. In Across the Sea of Stars, Harcourt, Brace, New York, 1959.
A humorous science-fiction story of war lost by the side with better technology, because they kept changing it.

DUF80A Duffy, N. D., Assad, M. G., Information Management -- An Executive Approach, Oxford University Press, Cape Town, South Africa, 1980.
Chapter 2 (Informations Systems Development Life Cycles) characterizes strategies as Linear, Loopy Linear, Plug-in, and Prototype, then offers attributes that may guide choice of strategy for a given project.

DUN80A Dunnigan, J. F. The Complete Wargames Handbook. Morrow, New York, 1980.
Survey of serious games, which often resembles real life in aspects of strategic planning, allocation of scarce resources, and risk analysis.

FAL81A Fallows, J. National Defense, Random House, New York, 1981.
Superb analysis of U.S. military weapons procurement and realities. Chapters: Realities, Managers, Magicians, Two Weapons, Employees, Theologians, Changes. "Two Weapons" chapter: F-16 fighter, brilliantly designed with good analysis and prototyping; then made less capable by adding things in opposite direction of original design philosophy.

GOM82A Gomaa, H. The Impace of Rapid Prototyping on Specifying User Requirements, in Proceedings ACM SIGSOFT Software Engineering Symposium: Rapid Prototyping, Columbia, Md, April 19-21, 1982, paper #14.
Case study of project done at GE.

JON80A Jones, A. J. Game Theory: Mathematical Models of Conflict, Wiley, New York, 1980.
Chapter 1 is reasonable introduction to fundamental concepts and analysis techniques (decision trees, minimax, etc).

KEE81A Keen, P. G. W. Information Systems and Organizationsal Change.

Comm. ACM 24, 1 (Jan 1981), 24-33.
Discusses long-term change in organizations related to information systems. Reviews causes of social inertia, resistance, and counter-implementation. Good reading for any software designer who hopes to sell ideas. Includes good bibliography.

LEH80A Lehman, M. M. Programs, Life Cycles, and Laws of Software Evolution. Proc. IEEE 68, 9 (Sep 1980), 1060-1076.
S-, P-, and E-programs; use of methods to predict release effects.

LUC57A Luce, R. D., Raiffa, H. Games and Decisions -- Introduction and Critical Survey, Wiley, New York, 1957.
Chapters 1, 3, 4, 7, 8 form a reasonable introduction to the topic.

MAC68A Macksey, M. C. Panzer Division -- The Mailed Fist. Ballantine Books, New York, 1968.
History of German tanks in WW II.
Moral: when you're even or ahead, don't stop, somebody is gaining on you. When you're behind, be careful how you try to catch up.

MAR82A Martin, J. Applications Development Without Programmers. Pretice Hall, Englewood Cliffs, NJ, 1982.
Illustrates existing methods and tools for 10- to 100-fold productivity improvements over conventional methods, for some problem domains. In some cases, normal high-level language coding appears to be pathetically absolete.

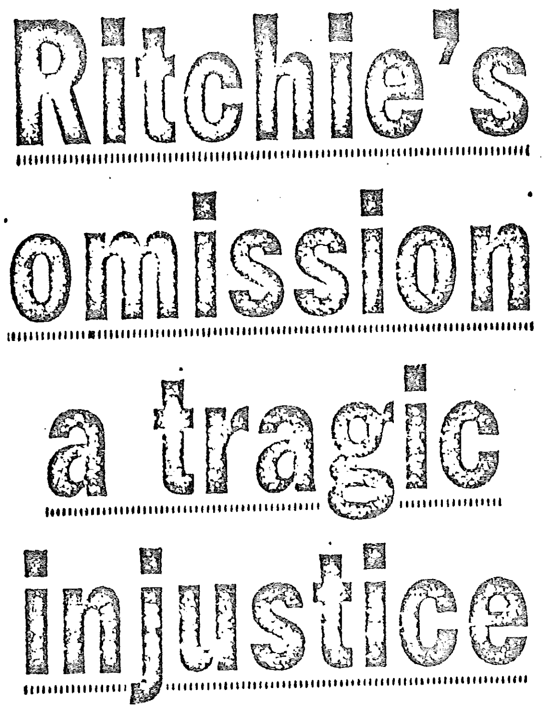
PYS82A Pyster, A., Boehm, B. W. The Impact of Rapid Prototyping on Software Development Standards. In Proc. ACM SIGSOFT Software Engineering Symposium: Rapid Protoyping, Columbia, MD, April 19-21, 1982, paper #28.
TRW builds Software Productivity System on UNIX with fast prototyping.

TAY82A Taylor, T., Standish, T. A. Initial Thoughts on Rapid Prototyping Techniques. In Proc. ACM SIGSOFT Software Engineering Symposium: Rapid Protoyping, Columbia, MD, April 19-21, 1982, paper #40.
Some parts are a bit academic, but has some good overall thoughts on prototyping, especially the section "Limits of Prototyping" on page 13.

TZUXXA Tzu, Sun. The Art of War. 500 B.C. Military Service Publishing Company, Harrisburg, PA, 1944. [thanks to L. Bernstein]
Contains numerous pithy discussions of project management, although not expressed in the standard terminology.
"When you engage in actual fighting, if victory is long in coming, then men's weapons will grow dull and their ardour will be dampened.... Thus, though we have heard of stupid haste in war, cleverness has never been associated with long delays."

"[Frederick the Great, in his Instructions to his Generals, says 'Those generals who have had but little experience attempt to protect every point; while those who are better acquainted with their profession, guard against decisive blows at decisive points, and acquiesce in smaller misfortunes to avoid greater.' In other words, keep away from sideshows.]"

WEI82A Weiser, M. Scale Models and Rapid Prototyping. In Proc. ACM SIGSOFT Software Engineering Symposium: Rapid Protoyping, Columbia, MD, April 19-21, 1982, paper ≠42.
Offers clear characterization of 3 different kinds of prototypes: user interface, functionality, and performance.



Ritchie's omission a tragic injustice

DAILY TELEGRAPH, Wednesday, December 8, 1982

Date: 19 Nov 1982 1655 (Friday)
From peteri:usa Thu Nov 18 15:42:00 1982
To: kev chris piers davec
Subject: system V

System V was announce on Tuesday, publically. Release 1st quarter next year. "AT&T will actively assist licensees with serveral levels of software support including hotline service, consultation, technical seminars, newsletters, electronic mail to report problems and periodic releases to correct problems.

"this new release reflects our response to users of UNIX systems who have indicated a desire to have the most current version available to the.....

"Software support also addresses concerns which system users have raised in the past"

"UNIX System V will be offered as a machine-independant standard operating system for 16-bit and 32bit computers. Details will be announced in January, 1983.

Future releases will be upward compatable with UNIX system V.

Bell/Western is now in the software business. Hope they make out.......

pete

From harpo!decvax!aps:usa Wed Feb 9 20:21:53 1983
To: harpo!mhtsa!australia!dave:csu40
Subject: DEC announcement of VAX UNIX

Basically, DEC has announced that it will sell and support UNIX on the VAX line of computers. (In December at DECUS, DEC announced that it would support V7M-11 on its PDP-11 line of computers. V7M-11 comes from the older DEC distribution V7m which is based on V7.) Before the end of the year, DEC will start selling and supporting a Berkeley (most probably 4.1) based VAX UNIX product with some of the features and things from System III/V that people seem to like or think they need (VPM, SCCS, etc). If 4.2 is at a point where it is stable before then (timming problems) DEC may go directly to 4.2 at the onset. No matter which BSD is used (4.1 or 4.2), DEC will go to 4.2 eventually. It is felt that we should wait until 4.2 has had time to settle out and "get the bugs out". Please feel free to pass this information on to others in Australia. Note that these offerings will be available only to the U.S. at first.

aps.

AUUGN is produced entirely by VOLUNTEERS from the Australian Unix Users Group. Unix related articles are solicited from readers. Please send your contributions to:

Bob Kummerfeld
Basser Department of Computer Science
University of Sydney
AUSTRALIA 2006

The subscription fee for AUUGN is $24 ($30 outside Australia) for six issues over one year. Please do not send purchase orders, only money! The volunteers running AUUGN don't like paper warfare so please treat this as an invoice if you wish. Your subscription fee should be mailed to:

Chris Rowles
SIROMATH
1 York St.
Sydney NSW
AUSTRALIA