EUUG

EUROPEAN UNIX® SYSTEMS USER GROUP NEWSLETTER

Volume 5, No. 2
SUMMER 1985

EUUG

European UNIX† Systems User Group

Newsletter Vol 5 No 2

(Not much of a) Summer 1985

† UNIX is a Trademark of Bell Laboratories.

Two Unique References — for UNIX™ Users

Price £16 Price £7.50

Hundreds of entries • Valuable technical information • Addresses of manufacturers • Quick reference indexes • Easy-to-follow classified entries.

If you want to get the latest information on UNIX hardware, software and service, then order your copies NOW!

Contact:

The Secretary, EUUG, Owles Hall, Buntingford, Herts, England SG9 9PL.
Tel: 0763 73039

Yet Another Implementation of Coroutines for C

Theo de Ridder

HIO department
IHBO "de Maere"
7500 BB Enschede
Netherlands

(ridder@im60.UUCP)

ABSTRACT

That the good old simulation package of Simula67 still offers an adequate test-set for completeness and usefulness of a coroutine implementation is demonstrated with a simple but effective machine independent scheme for C.
A single 'spaghetti' stack is used to obtain efficiency and compatibility with the C function mechanism. The preprocessor and datastructure facilities of C are combined into a general framework for an object-oriented programming style.

1. Introduction.

This article is a reaction to a recent publication of Bailes [1] about a low-cost implementation of coroutines for C. At first sight his approach looked promising but closer examination revealed a number of severe drawbacks. While many words were spent on a classification of the language extension, the basic choices were not argued at all. More specific objections concern:

a) Because stackframes for coroutines were considered as permanent objects, it was quite unclear why the heap was needed for saving environments.

b) A single stack was used for compatibility with C. Any C function could be called from a coroutine but the reverse was not true. Only coroutines were allowed to invoke coroutines. This is an unnecessary and unacceptable limitation. It makes it impossible to solve any non-trivial application in a modular way. Apparently Bailes never saw the famous article of Bobrow and Wegbreit [2] about using a single stack for multiple environments.

c) While Simula67 was quoted as a starting point, the essence of the class concept was bypassed completely.

The fact that C is a widely used language makes it relevant to offer a much better solution, which will be described with only C as the description language, to enable anyone to use the same scheme in other environments.

2. Coroutines.

The two context switching primitives SavEnv() and RstEnv() are given in program 1. The machine dependent register save and restore will force them to be transposed into assembler code.

The semantics of SavEnv() are very close to the fork() system-call of UNIX. A boolean return-value discriminates between the parent and child. A child might return, but then parent and child not only share the same code but also the same stackframe. This enables a facility to free stackspace which should be applied with extreme care.

```
typedef struct corout {
    Word            cregisters[WREGS];
    Stackframe      *csf;
    Instruction     *creturn;
    struct corout   *cparent;
} Corout;

extern Stackframe   *sf;    /* see program 2 */
extern Corout       *CurCor;

#define NEW(corp, call) \
        if (SavEnv()) {\
            (corp)->cparent = CurCor;\
            CurCor = corp;\
            call; }

#define RESUME(corp) \
        if (SavEnv()) {\
            CurCor = corp;\
            RstEnv(CurCor); }

#define DETACH      RESUME(CurCor->cparent)

Bool SavEnv()
{
    sav_regs(CurCor->cregisters);
    CurCor->csf = sf->prev_sf;
    CurCor->creturn = sf->prev_pc;
    return(TRUE);
}

Bool RstEnv()
{
    rst_regs(CurCor->cregisters);
    sf->prev_sf = CurCor->csf;
    sf->prev_pc = CurCor->creturn;
    return(FALSE);
}
```

Program 1.

Having the primitives available, the real problem still has to be solved: maintaining a single stack for the non-LIFO context switches. The general solution of [2] will be simplified for our specific situation.

The first thing one has to realise is that most C compilers produce code which strongly couples the stackpointer (sp) with the current stackframe (sf) in the case of function entry or exit. The basic goal is to guarantee sp-invariance after function calls. This of course is no longer possible for functions leading to a resume of a coroutine. To be able to use a single stack one has to break the automatic sf-sp coupling. The solution of never giving back any stackframe is too unrealistic. By simply maintaining a last-framepointer (lf) it is possible to free any unused frame appearing on the top of the stack. At any moment the expression `(lf == sf)` indicates if the current stackframe is the last one that is allocated. Assuming that an architecture with 0 as a possible return-address will be very rare, this value is used to mark unused stackframes.

A second problem for coroutines in C is the fact that actual parameters of any function are released from the stack after the call, which again implies sp-invariance of the call. In our case actual

parameters may dropped only if `(lf == sf)` holds.

In program 2 all the relevant new extra code for entering, leaving and calling a function is given. The stack is assumed to grow downward. A local variable is accessed by `sf->locals[-index]`.

```
typedef struct stackframe {
    Stackelement        locals[];
    struct stackframe   *prev_lf;
    struct stackframe   *prev_sf;
    Instruction         *prev_pc;
} Stackframe;

Instruction  *pc;    /* program counter */
Stackelement *sp;    /* stackpointer */
Stackframe   *sf;    /* current stackframe */
Stackframe   *lf;    /* last allocated stackframe */

entrycode(nlocals)
int nlocals;
{
    /* prev_pc is filled by jsr(label) */
    push(sf);
    push(lf);
    sf = (Frame *)(sp - 1);  /* include first local */
    lf = sf;
    sp -= nlocals;
}

exitcode()
{
    Instruction *newpc;
    newpc = sf->prev_pc;
    sf->prev_pc = 0; /* mark unused frame */
    if (sf == lf)
        while (lf->prev_pc == 0)
        {       /* remove unused frame */
            sp = (Stackelement *)lf;
            lf = lf->prev_lf;
        }
        sp += (Sizeof (Stackframe)) / (Sizeof (Stackelement));
    }
    sf = sf->prev_sf;
    pc = newpc;
}

callcode(nargs, label)
int nargs;
Instruction *label;
{
    push_args(nargs);
    jsr(label);
    if (sf == lf)
        drop_args(nargs);
}
```

Program 2.

The final problem is when the given code should be applied. Code for functions which do not cause any resumption of a coroutine before returning can remain unchanged. This fulfills the basic requirement that any existing compiled library can be used as before. On the other hand, does it matter if extra code is produced for functions not invoking coroutines? It is a complication, but it does! When function-calls are used in (nested) expressions, the compiler again assumes sp-invariance, and therefore actuals must be freed.

It may criticised for being simplistic, but we chose to implement the following solution. Any function which may cause a coroutine switch starts its name with an underscore. Any file containing such functions is compiled with the -S option into assembler code. A simple postprocessor written in awk [3] is then used to change the code at all relevant places.

To prevent programming errors all underscore_functions should have type `Void`. Another requirement is that those functions should start with an dummy local if there is no redundant place in a stackframe available for the prev_lf link.

3. Classes.

As a language we can regard C [4] as a more or less grown-up structured assembler. Once upon a time in its childhood it got structures. Growing up, the names of attributes even became local. Becoming adult C longs for classes [5], and maybe before it dies it will become as wise as Simula67 was already in the old days.

However, to be practical one has to compromise, and sometimes the results are not too bad. With type casting and preprocessor facilities many aspects of object-oriented programming can be approached. One of the basic limitations is of course the absence of many compile-time checks. In program 3 the fundamentals of a full implementation of the Simula67 simulation class [6] are shown. From this program it will become clear that the reason to avoid implicit core-allocations for new coroutines or subclasses is to get all data of a class instance in a single C-structure.

```
typedef struct link {
    struct link  *pred;
    struct link  *suc;
    Bits         classes; /* memberships */
} Link, Head;

typedef struct event {
    Link             link;
    Real             evtime;
    struct process   *proc;
} Event;

typedef struct process {
    Link         link;
    Corout       corout;
    Bool         terminated;
    Event        *event;
} Process;

typedef struct simulation {
    Process      main;
    Head         sqs;
} Simulation;
```

```
#define CLASS(me, type) \
        if (!(((Link *)me)->classes & type))\
            ERROR("classtype");

#define NEW_LINK(L) (L)->classes = LINK;

#define NEW_HEAD(L) (L)->classes = HEAD;\
        (L)->pred = (L)->suc = L;

#define NEW_EVENT(E, T, P) \
        E = (Event *)ALLOC(sizeof (Event));\
        E->link.classes = (LINK|EVENT);\
        E->evtime = T; E->proc = P;

#define NEW_PROCESS(S, P, funname) \
        (P)->link.classes |= (LINK|PROCESS);\
        (P)->event = NONE;\
        (P)->terminated = FALSE;\
        NEW(&(P)->corout, BODY(S, funname));

#define BODY(S, funname) \
        DETACH;\
        funname(Current(S));\
        (Current(S))->terminated = TRUE;\
        _Passivate(S);\
        ERROR("resuming terminated process");

#define NEW_SIMULATION(S, funname) \
        (S)->main.link.classes |= SIMULATION;\
        NEW_PROCESS(S, &(S)->main, _MainMain);\
        NEW_HEAD(&(S)->sqs);\
        NEW_EVENT((S)->main.event, 0.0, &(S)->main);\
        Into((S)->main.event, &(S)->sqs);\
        funname(S);\
        _Hold(S, 100000.0);
```

Program 3.

One of the main differences with the original code is due to the inability of the preprocessor to define new infix operators. Each function associated with a certain class is given an extra argument as a self-reference to the involved instance of the class. As an example the code for the procedure `_Passivate` is shown in program 4. It will be left as a self-assessment exercise for the reader to analyse why, within the macro `BODY` of program 3, replacing the call of `_Passivate` by its body will in general save a lot of stackspace.

```
Void _Passivate(me)
Simulation *me;
{
    Process *pr;
    CLASS(me, SIMULATION);
    pr = Current(me);
    Out(pr->event);
    FREE(pr->event);
    if (Empty(&me->sqs))
        ERROR("sqs is empty");
    pr = Current(me);
    RESUME(&pr->corout);
}
```

Program 4.

There exists an old simulation example [7] concerning a building with a lift. With the datastructures and above programming style, the original code (after removing some bugs!) can be taken over straightforwardly to give program 5.

```
typedef struct floor {
    Link        link;
    Int         number;
    Head        queue;      /* people waiting */
    Bool        upbutton;
    Bool        downbutton;
} Floor;

typedef struct lift {
    Process     process;
    Head        load;       /* people inside */
    Int         position;
    Status      state;
    Direction   way;    /* moving direction */
    Bool        buttons[N_FLOORS];
} Lift;

typedef struct person {
    Process     process;
    Int         ident;
    Int         at;     /* coming from */
    Int         to;     /* going to */
    Real        arrivaltime;
    Real        maxwaittime;
} Person;

typedef struct building {
    Simulation  sim;
    Floor       floors[N_FLOORS];
    Lift        lift;
} Building;

extern Void _LiftMain();
extern Void _PersMain();
extern Void _BuildMain();
```

```
#define NEW_FLOOR(F,N) \
        NEW_LINK(&(F)->link);\
        NEW_HEAD(&(F)->queue);\
        (F)->number = N;

#define NEW_LIFT(S,L) \
        NEW_HEAD(&(L)->load);\
        (L)->process.link.classes = LIFT;\
        NEW_PROCESS(S, &(L)->process, _LiftMain);

#define NEW_BUILDING(B) \
        &(B)->sim.main.link.classes = BUILDING;\
        NEW_SIMULATION(&(B)->sim, _BuildMain);

#define NEW_PERSON(S,P,ID,AT,TO,AR,WA) \
        P = (Person *)ALLOC(sizeof (Person));\
        P->ident = ID;\
        P->at = AT;\
        P->to = TO;\
        P->arrival = AR;\
        P->wait = WA;\
        P->process.link.classes = PERSON;\
        NEW_PROCESS(S, &P->process, _PersMain);


/* writing a class body goes like _BuildMain */

Void _BuildMain(me)
Building *me;
{
    int i;
    Floor *f;
    CLASS(me,BUILDING);
    for (i=0; i < N_FLOORS; i++)
    {
        f = &me->floors[i];
        NEW_FLOOR(f, i);
    }
    NEW_LIFT(&me->sim, &me->lift);
    ACTIVATE(&me->sim, &me->lift);
    for (i=1; i <= Population; i++)
    {
        Person *p;
        NEW_PERSON(&me->sim, p, i, RANDOM1,
            RANDOM2, Time(&me->sim), NORMAL);
        ACTIVATE(&me->sim, p);
        _Hold(&me->sim, NEGEXP);
    }
}
```

Program 5.

4. Conclusions.

The given scheme of using single stack for a coroutine implementation is very efficient and easy to implement. The price of simplicity is a certain waste of space and an unstable stack in case of mutual incursive coroutine loops. A real general solution will need a garbage collector and multiple stacks or a stack frame copying mechanism.

For practical applications one should combine coroutines and datastructures into a single concept like classes.

We know that the given solution for such an object oriented programming style is just a poor-man's approach. However, it might be attractive to others because recent literature and network messages reveal a lot of poverty within the UNIX world.

References

1. P.A.Bailes, A Low-Cost Implementation of Coroutines for C, Software-Practice and Experience, 15, 379-395(1985).
2. D.G.Bobrow and B.Wegbreit, A Model and Stack Implementation of Multiple Environments, Communications of the ACM, 16, 591-603(1973).
3. A.V.Aho, B.W.Kernighan and P.J.Weinberger, Awk - A Pattern Scanning and Processing Language, Software-Practice and Experience, 9, 267-279(1979).
4. B.W.Kernighan and D.M.Ritchie, The C Programming Language, Prentice-Hall, 1978.
5. B. Stroustrup, Data Abstraction in C, AT&T Bell Laboratories Technical Journal, 63, 1702-1732(1984).
6. O-J.Dahl, B.Myhrhaug and K.Nygaard, The Simula67 Common Base Language, Publication S-22, Norwegian Computer Centre, Oslo, 1970.
7. H. Rohlfing, Simula: Eine Einführung, Bibliographisches Institut AG, Mannheim, 1973.

C++

One of the latest technical developments from AT&T is the programming language C++. This is a superset of the C programming language; it is fully implemented and has been used for non-trivial projects. There are now more than one hundred C++ installations.

Features of the language include Simula-like classes providing (optional) data hiding, (optional) guaranteed initialization of data structures, (optional) implicit type conversion for user-defined types, and (optional) dynamic typing; mechanisms for overloading function names and operators; and mechanisms for under-controlled memory management. New data types, like complex numbers, can be implemented, and "object-based" graphics packages can be structured. A program using these data abstraction facilities is at least as efficient as an equivalent program not using them, and the compiler is faster than older C compilers.

C++ is available to Educational Institutions, from UNIX Europe Limited for a fee of £150. Please apply to Mr. G. Edwards at the following address for further details:

Unix Europe Limited,
27A Carlton Drive,
LONDON SW15 2BS.

Yet Another Paris Report

Sebastian Schmidt

snoopy@ecrcvax.UUCP

Disclaimer:

All opinions herein are my own and not my employers or anyone elses. To be read at your own risk.

This is a **very informal**report It was my very first EUUG conference ever and I thouroughly enjoyed it.

Lots of interesting people with a wide range of good talks and the right measure of wit to keep everyone awake. It was very well organised - talks finished on time and everyone stuck to the schedule (+/- 5minutes). What a change to DECUS in Munich which was crap by comparison. Just shows how much better enthusiastic individuals can be rather than a multi-million dollar company. Funny thing is that multi-million dollar companies seem to start out as a bunch of enthusiastic individuals but somehow seem to lose their organisational talent very quickly.

I met very interesting people and old friends amongst them Peter Langston (formerly of Lucasfilm now Bell Research), Peter Collinson (whom we had to restrain from showing us slides and videos of his son), David Rosenthal (who gave some very interesting tips regarding our ROSE project), Jenny Martin from Acorn (with company - she will fork() soon), Tim Snape (a.k.a. Jesus) ex. QMC with some NEW jeans (would you believe it) and Phil Lander (ex. QMC) who is now the Nick/Snoopy clone (i.e. guru for Unix and hardware) for the AEC in Winfrith. Also met Bip who is Ash Dattani's brother and a great time was had by all of us.

Simultaneous translation was provided but the translators were regularly pissed off because people spoke at 19,200 Baud or didn't work the mikes properly. This of course produced more laughter (for the people wearing headphones) as time went by.

Anyway lets go down the Agenda. Due to flight planning I missed the opening hoo-ha and only arrived in time as Ian Johnstone held a talk on a multi-processor UNIX system. The idea was to have a flexible system so that in case of CPU's dying the machine would still run and reconfigure itself. For this they introduce this new bus called System Link Interconnect and Control (== SLIC). They have actually built a SLIC chip and the advantage is that you do not waste any bandwidth of the data bus. Performance for a multi processor system is very close to the theoretical linear limits and it seems to be only limited by the discs (what else is new?).

Then a talk on concurrent processing in Ada and UNIX. Basically this described the different philosophies behind Unix and ADA (yuk!) and it showed that the two approaches (Ada tasks and Unix processes) were very different. Also Unix pipes are a limited way of inter-process communication. So Henk decided to leave his fingers off this. It just doesn't seem to fit.

Then MIRANDA was introduced. This is a new functional programming language by David Turner (who looks like a slimmed down version of Samson Abramsky without a beard which caused a good deal of speculation if there is a looks-linked chromosome which specifies the individuals functional programming ability (lazy evaluation of genes?)). It's got all the goodies (lazy evaluation, polymorphic typing etc.). It looked nice and they are working on a VLSI implementation to increase the speed. Building your own chips seems to be the latest fashion/rage/cult in the US and the wave is coming over to Europe at an amazing speed. I liked it because it isn't like LISP (i.e No Lots of Irritating Silly Parantheses).

After this morning session the afternoon one followed with a really fun talk by Peter Langston, where he introduced his latest video games called Ball Blazer (with music by Pat Metheny!!) and Rescue on Fractalus‡ The videos were great and got everyone into a high and over-sexed mood. They really were well worth the entrance fee. Its amazing what you can get out of an Atari 800. (More than a QMC VAX 750 har, har, har). Peter has an amazing sense of humour and his speech was a blast.

After this Tom Duff had a tough act to follow. However he also had laughter on his side when he presented his overhead slides. They were positively the worst slides I have ever seen. Characters at any orientation and words varying in colour as they meander across the foil. I nearly wet myself with laughter and could only manage cries like "Give the man a keyboard!". Contrary to what I thought I was not evicted for this and when I managed to get the tears out of my eyes he actually gave a very interesting talk on 3-d images under Unix. He has developed a few algorithms and showed us a little movie where flying saucers intersect the ground at arbitrary planes just to show that his algorithm handles several planes correctly. It was very interesting to see how the film he showed would have represented the state of the art about 5 - 10 years ago (so he said) but was only about a weeks work these days.

Incidentally coming back to Tom's slides: there was a German delegate at the conference who protested about the appalling quality of the slides with words to the effect: "This should not happen at such an IMPORTANT conference" (my capitals). Well I never. Does that not just take the biscuit. EUUG an IMPORTANT conference. Apart from that I liked the slides - they had character and style (albeit a very individualistic one) and they certainly accomplished two things:

1) They got all the info across

2) They amused as well (if even involuntarily)

3) Oh yes: they showed that some Germans lack humour. Je reste mon cas.

Then a french talk by Etienne Beeker which featured image synthesis under Unix. He makes inserts for TV and his colour imagery was very, very good. As a matter of fact his slides showed amazing realism in computer generated images.

I then had to leave so I missed the Birds of a Feather session (the Unix way of saying special interest groups).

The next day was kicked off by a talk by Patrick Bellot about VIDMAN which is a system which supports documentation generation. It maintains a hierarchially structured document type (have you had a look at Scribe lately - ho hum).

Then a good one: David Rosenthal from CMU and he told us all about his network which has about 600 machines on an Ethernet and will have about 8000 nodes soonish (next year or so). So this man has dealt with a lot of problems and has given me some good tips about how we best go about the implemantation of the ISO stuff. We may have a lot of difficulty with this. I have his phone and address and he seems to be willing to help us with stuff.

The CMU network is really amazing and he showed us some examples of cut&paste between two machines on the network with really snappy response - like a single user perq. They have plenty of Suns and have this amazing way of running scribe all the time in a sort of raw mode which means that stuff on your screen always gets centered (if you are in a centered environment) AS YOU TYPE! This is the way to go. I loved it. But then I guess our local scribe users will have different opinions.

In the post-refresh and pre-lunch session Radek Linhardt gave a glowing appraisal of X.25 as opposed to the phone system. It gave a really good overview. X.25 is the way for the EUNET as far as I can see. It offers the best compromise in the speed/security dilemna. We use X.25 here and we

---

‡Rescue on Fractalus and Ball Blazer are registered trademarks of LucasFilm Inc.
Unix is a reg. trademark of At&T Bell Labs.
VAX is a reg. trademark of Digital Equipment Corp.
Any other trademarks are acknowledged etc. etc.

are very happy with it.

Then something from down-under: ACSNET was introduced to us. It looked very good. Finally someone has trashed the oft-hacked uucp and come up with something clean and new and working. It was time. I feel we should try to get this as it supports a *much* nicer addressing scheme than uucp and could even resolve the controversy between the "bangists" and the "domainists" (Is there a controversy really?). I think everyone felt that it is much nicer to have domains than to have to type dem 25 site adresses. Peter Collinson has expressed an interest in the ACSNET software and CWI already have the tape. Let's hope something comes out of this. This may be the chance to do away with some of the nasty uucp stuff.

Anyways the afternoon was still devoted to issues with mail: Steve Kille from UCL introduced us to the marvels of MMDF II, which takes the place of sendmail. It has to be able to deal with a whole bunch of addresses etc. This seems a good idea, however it seems to me that it will not really be a constituent of uucp or mail in the future.

The second session that day was taken up with the question "What is an international UNIX system?". What is fate? What is guilt? What is UNIX? Duncan Missimer pointed out some shady areas in UNIX where problems with the language do occur. basically there are problems like error messages and things like the format of the date etc. Then of course the problems of character sets. These things are not easily solved.

I did not go to the later sessions that day because I wasn't particularily interested in a chinese UNIX or Greek characters. I spent the time at the concurrently running exhibition. This was a small affair but I had a lot of fun playing with SUNS, picking up freebies (inc. bottles of 1986 Chateau NUXI du Pape Appelation Uncontrolee) and chatting with the chaps from The Instruction Set. However nothing really new in the way of hardware was to be seen. Not a soul at the Zilog booth so I guess they are really way out of the race.

Oh yes. Phil was along at the exhibition and at the Sun booth he claimed that one of the machines there was the one he ordered, which Sun said "has been delayed in customs". He looked at the serial number and we had to forcefully restrain him from forcefully dismantling the nearest discless node in order to get at his Sun...

We spent about an hour or so racing up and down the Champs-Elysee taking pictures like crazy and admiring Pierre Cardin shoes which cost about #250 and believe you me they did not look at all special.

Then in the evening the conference dinner on the Bateaux Mouches, which are funny things: they have big lights illuminating the banks of the river, which must make it hell living in a flat along the Seine. However seeing the sights of Paris was lovely and the dinner was very good. We had a great time later on the top deck waving at pretty Parisian girls and Phil set back the diplomatic relations with Japan about 10 years by chatting to some Japanese honeymooners. So a great time was had by all.

After the trip a walk back to the hotel, where we tried several bottles of the aforementioned Chateau NUXI while watching this amazingly boring french horror movie on TV. (Avec sous-titres, sans doute!)

Pretty soon the conversation centered around horrible things like "The virtues of 4.2 vs. VMS" (None really we agreed), "How to swap out 4 GB of stuff and is it really necessary?" (Obviously no - you never have physical 4 GB on a VAX) "How to make your silly VAX boot after you have stuck those EL-cheapo memory boards in it". As said a very boring movie but it gave us ideas of what to talk about.

At around 2:30 am the meeting dissolved along with my liver, which is the time most peoples metabolisms seemed to have given up for on the way back to my hotel I saw a good deal of familiar faces trying the revolving doors of the Meridien. Incidentally these doors revolve all the time (no human power necessary) and the sections are big enough for a busload of people. Radio 4 listeners please note!

The next day (Oh god, could you please turn down the PA amplifier!) featured something known only to adherents of DECUS meetings - a change of agenda! A birds of a feather session re. a standard UNIX which wasn't very interesting for me at least so I hijacked David Rosenthal and twisted his arm until he gave me some pointers of how to put new network protocols into the 4.2 kernel. Aaaaaah, the sheer joy of hacking! Anyway this jam session took about an hour and very useful it was indeed. Good that I did not come back to Munich early.

Then before lunch two more talks: the first on a contractual mode of software development with some very elucidating comments re. the inside of Magaret Thatcher's head and the internal workings of GEC. The talk basically was a hype of top-down design. I feel that this is not reality: real programmers don't do top-down or bottom-up: they do middle-out. This natural occurence was not remarked upon. Too bad. Thats what's likely to screw up the nice model. The real world is not a model.

The last talk I attended was the automatic generation of make dependencies. This was a good one but in the end only boiled down to "read my article in so and so...". This was a shame but couldn't be helped. I would have loved to hear more.

I had to give the afternoon debates a miss because I had to catch my plane back home: time for the good byes and see you in Copenhagen.

Summary:

1) Excellent conference
2) Great people
3) Good talks
4) It was great. I will go again. Even if I have to pay myself!

Love,
Snoopy.

PS

Ok, since people said I should make my conference write-ups even longer here is a bit which I only wanted to include in the previous write-up for the girls at QMC:

Fashion Report

On the way out is the neo-romantic/Boy George look for girls. Very few of those still around. Basically fashion seems to be tending towards more pronounced feminine concepts. No more of that unisex look where you have to be very well into the introduction game to find out what sex your partner really is (as if it mattered).

Patterned black stockings seem to be all the rage so I got about a dozen.

Men seem to adhere to an older style i.e. the Alphaville look. This is the only way to describe it although I saw quite a few Shakin Stevens look-alikes (yeeechhh!). So rockabilly is still very much alive there. That put me off somewhat.

UK UNIX® Systems User Group

Owles Hall,
Buntingford,
Herts. SG9 9PL.
Tel: 0763-73039
Facs: 0763-73255
Network Address:
mcvax!qtlon!euug!euug

CALL FOR PAPERS

One Day Technical Meeting

MONDAY 16th December 1985

Oliver Thompson Lecture Theatre, City University, London

Program Chairman: Sunil K. Das, UK UUG Chairman

Suggested topic areas include, but are not limited to:

* Standards (UNIX and C)
* Software Tools (YACC, LEX, new shells, AWK,...)
* Networking, Mail, News and UUCP
* Alvey, Esprit and Common Base Policy
* Future Directions (System V.x, BSD4.x)

Abstracts should include (for a 20-25 minute talk)

* Title of Presentation
* Author(s) name(s), affiliation(s)
* Address (with electronic mail address, if available)
* Telephone number (and telex, if available)
* Special requirement

Abstracts should be submitted by 1st November, 1985 to:

UK UNIX Systems User Group

Sunil K Das
Chairman

The City University,
Computer Science Dept.,
Northampton Square,
London EC1V 0HB.
Tel: (01)-253 4399

NETWORK ADDRESS:
uucp:ukc!ucl-cs!sunil

† UNIX is a Trademark of Bell Laboratories.

EUUG

European UNIX[TM] systems User Group

SPRING CONFERENCE AND EXHIBITION

Florence, Italy, 21–24 April 1986

CALL FOR PAPERS AND TOPICS

The next EUUG Conference and Exhibition will be held in Florence, Italy during the Spring of 1986.

Technical Sessions
The Technical Programme will run for three days from 21–24 April. The main theme of the Technical Conference will be real applications of the UNIX system. This is a direct follow on from the X/OPEN group announcement made at the Copenhagen Conference. Notwithstanding this, papers on all subjects relating to the UNIX system will be considered.

Papers and suggestions for topics are urgently solicited.

Exhibition
The EUUG has decided to run two conferences per annum, one of which will host the major European UNIX exhibition. At the Florence Conference, there will be a 3-day exhibition opening on 21 April.

Tutorials
Following the tremendous success of the Tutorial Sessions at Copenhagen, there will be a day of tutorials addressing advanced features of UNIX. This will be held on 21 April with a possible extension into 22 April if demand necessitates. All tutorials will be presented by experts who are recognised in their field. A comprehensive set of notes will be distributed with each tutorial.

Suggestions for topics appropriate to your individual needs are required.

Industrial Sessions
Also, on the first day of the Conference before the start of the Technical Programme, we will offer an industrial component. Papers for this are required along with ideas for subjects that should be addressed. These talks are intended to be commercial and not necessarily technical in content: they should if possible appeal to all the attendees.

General
If you are willing to present a paper at any of the sessions mentioned above, or indeed if you would like to suggest an area of interest to you or your organisation, please complete the form below and return it to the address indicated as soon as possible.

Deadlines for submission are as follows:

Suggested topics:	immediate
Formal abstracts:	1 December, 1985
Complete papers:	20 February, 1986

If you wish to receive booking information for the Conference when this is available, please contact the EUUG immediately. Pre-Conference bookings will be taken and will offer a significant discount on the full price.

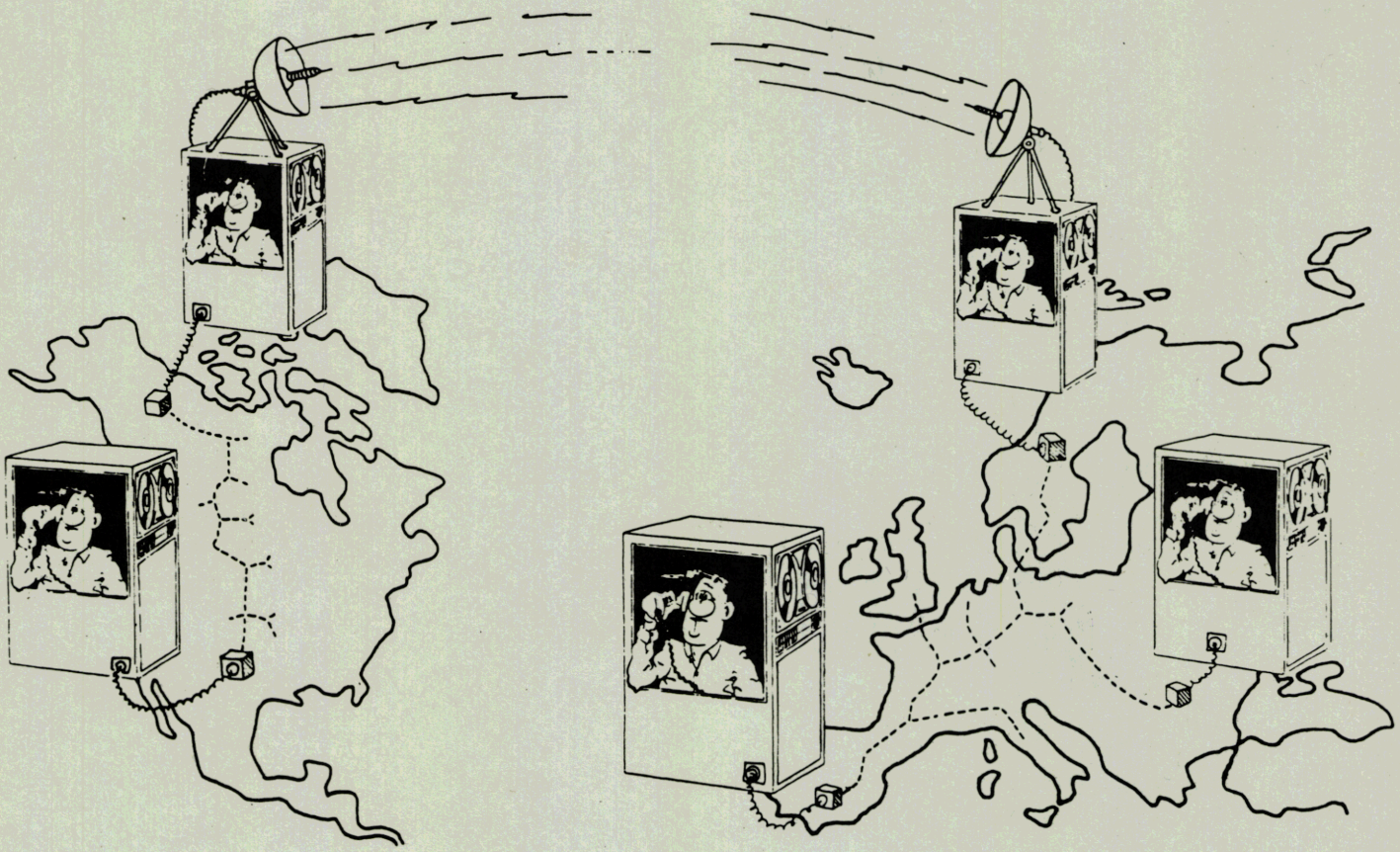
EUUG Florence 1986

Name: .....
Address: .....
UUCP: mcvax! .....
Telephone: .....

Topic: .....
Suggested speakers: .....
(if not yourself)
Comments/ideas: .....

Contact (to whom this form should be returned):
Mrs Helen Gibbons, EUUG, Owles Hall, BUNTINGFORD, Herts, SG9 9PL, United Kingdom.
Tel: +44 763 73039

Programme Chair: Nigel Martin, The Instruction Set, 152–156 Kentish Town Road, London, NW1 9QB, United Kingdom. Tel: +44 1 482 2525 UUCP: mcvax!ukc!inset!nigel



The Secretary
European Unix® Systems User Group
Owles Hall
Buntingford, Herts.
SG9 9PL.
Tel: Royston (0763) 73039