Contents

If you need Help

1 General
   1.1 Letter from the Editor ................................. 1
   1.2 User (Account) Registration .......................... 2
   1.3 ASIS Anonymous ftp Server ........................... 2
   1.4 Optimizing Fortran compiler available for Solaris 2.x—free .......................... 2

2 VM to UNIX Migration Task
   2.1 The Proposed SP2 Central Computer .................. 3

3 UNIX Workstations and Desktop Support
   3.1 The Zephyr Message Service for Unix Users ........ 5

4 Communications and Networks
   4.1 The Recabling of CERN ............................... 6
   4.2 CERN Networks Status Available via WWW .......... 7
   4.3 Running DECnet/OSI (Phase V) at CERN .............. 7

5 Text Processing
   5.1 Text Processing: An Update .......................... 8
   5.2 "xwpick" New Version of "xpick" (Pick Images on an X11-screen) ......................... 8

6 Tutorial Section
   6.1 Interfacing Fortran and C ........................... 9

7 Program Library News
   7.1 Release 94B of the CERN Program Library .......... 16
   7.2 A Second Life of the CERN POISSON Program Package ................. 18
   7.3 End of my Activities with CERNLIB .................. 20
   7.4 PAW Release (2.05) .................................. 20
Editorial Notes

Past experience shows that it is a good policy to have four CNL editions per year (one around Easter, one before and one after the summer, and the last one just before Xmas), unless the number of contributions makes it necessary to have a supplementary issue, or to cancel one. Thus,

the date of the deadline for contributions to the next issue of the CNL is:

<table>
<thead>
<tr>
<th>Issue number</th>
<th>Final date</th>
</tr>
</thead>
<tbody>
<tr>
<td>218</td>
<td>Monday, 5 December 1994</td>
</tr>
</tbody>
</table>

Contributions to the CNL are accepted as plain text, although marked-up text in LaTeX is preferred. Articles, news items and letters intended for publication in the next issue should be sent directly to the editor (cnl.editor@cern.ch) without particular controls on the mail subject line.

The opinions expressed in this newsletter are those of the contributors and are not necessarily those of the CERN management. The editorial board reserves the right to edit, omit or hold-over copy due to lack of space.

This document was produced with LaTeX and the cernlnl style. Compressed PostScript files, containing the complete printable version of this CNL or parts of it can be obtained by anonymous ftp to asisftp.cern.ch as follows (commands to be typed by the user are underlined):

```
ftp asisftp.cern.ch
Name (asisftp:username): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: yourusername@yournode.domain.country
ftp> cd cnl217
ftp> binary
ftp> get cnl217.ps.gz (or get cnl217.ps)
ftp> quit
```

Please note that, if you do not have the gnu gunzip utility on your system you can get the uncompressed PostScript Version by typing the command get cnl217.ps, without the.gz suffix. In order to save Internet bandwidth, you are, however, strongly urged to try and install the gunzip utility since gzipped files are about three times smaller than their unzipped equivalents.

The following files related to the present CNL are available in that directory:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cnl217.ps.gz</td>
<td>Complete CNL</td>
</tr>
<tr>
<td>general.ps.gz</td>
<td>“General”</td>
</tr>
<tr>
<td>service.ps.gz</td>
<td>“UNIX Workstations and Desktop Support”</td>
</tr>
<tr>
<td>textproc.ps.gz</td>
<td>“Text Processing”</td>
</tr>
<tr>
<td>cforttut.ps.gz</td>
<td>“Interfacing Fortran and C”</td>
</tr>
<tr>
<td>edithelp.ps.gz</td>
<td>Editorial and help pages</td>
</tr>
<tr>
<td>vmunix.ps.gz</td>
<td>“VM to UNIX Migration Task”</td>
</tr>
<tr>
<td>network.ps.gz</td>
<td>“Communications and Networks”</td>
</tr>
<tr>
<td>proglib.ps.gz</td>
<td>“Program Library News”</td>
</tr>
</tbody>
</table>

We would like to encourage you to subscribe to the announcement of the PostScript version, a service which is becoming more and more popular. You just have to send a mail to cnl@cern.ch (the “CNL server” machine) with the subject line being:

Subject: ANNOUNCE POSTSCRIPT CNL

The CNL server will then send you a mail whenever there is a new CNL ready as a PostScript file.

To know more about what the CNL server can do for you, please send a mail to cnl@cern.ch with the subject line being:

Subject: HELP

On all central systems one has access to the terminal version of the CNL via the command:

```
XFIND CNL 217
```

On CERNVM XFIND will give access to the printable (usually PostScript) version as well.

Responsible editor: Nicole Cremel (e-mail: cnl.editor@cern.ch)  Technical realization: Michel Goossens
IF YOU NEED HELP (Contacts at CERN)

<table>
<thead>
<tr>
<th>Service</th>
<th>Name</th>
<th>Address</th>
<th>Telephone</th>
<th>E-mail Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER CONSULTANCY OFFICE (UCO)</td>
<td></td>
<td>513/R-052</td>
<td>4952</td>
<td><a href="mailto:user.support@cern.ch">user.support@cern.ch</a></td>
</tr>
<tr>
<td>Opening hours 08.30-12.00 and 14.00-17.30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All aspects</td>
<td>Miguel Marquina</td>
<td>513/1-020</td>
<td>4912</td>
<td><a href="mailto:user.support@cern.ch">user.support@cern.ch</a></td>
</tr>
<tr>
<td>Accounting Service</td>
<td>Attila Koppanyi</td>
<td>513/1-019</td>
<td>4933</td>
<td><a href="mailto:user.support@cern.ch">user.support@cern.ch</a></td>
</tr>
<tr>
<td>User Registration</td>
<td>M.C.Perler</td>
<td>513/1-017</td>
<td>5004</td>
<td><a href="mailto:user.support@cern.ch">user.support@cern.ch</a></td>
</tr>
<tr>
<td>Consultancy Office</td>
<td>R.Woolnough</td>
<td>513/1-020</td>
<td>9156</td>
<td><a href="mailto:user.support@cern.ch">user.support@cern.ch</a></td>
</tr>
<tr>
<td>Computer Documentation Office</td>
<td>M.Franceschi</td>
<td>513/1-022</td>
<td>4912</td>
<td><a href="mailto:user.support@cern.ch">user.support@cern.ch</a></td>
</tr>
<tr>
<td>CENTRAL COMPUTER OPERATIONS (24 HOURS)</td>
<td></td>
<td>513/R-066</td>
<td>5011</td>
<td>operator@cernvm</td>
</tr>
<tr>
<td>All aspects &amp; IBM</td>
<td>David Underhill</td>
<td>513/R-035</td>
<td>4920</td>
<td><a href="mailto:d.underhill@cern.ch">d.underhill@cern.ch</a></td>
</tr>
<tr>
<td>Central VAXes</td>
<td>Tim Whibley</td>
<td>513/R-033</td>
<td>4849</td>
<td><a href="mailto:t.whibley@cern.ch">t.whibley@cern.ch</a></td>
</tr>
<tr>
<td>Remote Printers</td>
<td>Remy Bouvy</td>
<td>513/R-004</td>
<td>4929,13+5548</td>
<td><a href="mailto:r.bouvy@cern.ch">r.bouvy@cern.ch</a></td>
</tr>
<tr>
<td>Networks</td>
<td>Alasdair Ross</td>
<td>513/R-034</td>
<td>4927</td>
<td><a href="mailto:netops@cern.ch">netops@cern.ch</a></td>
</tr>
<tr>
<td>CORE services - CSF</td>
<td>C.Boissat</td>
<td>513/R-031</td>
<td>3195,13+5630</td>
<td><a href="mailto:c.boissat@cern.ch">c.boissat@cern.ch</a></td>
</tr>
<tr>
<td>CORE services - SHIFT</td>
<td>Gordon Lee</td>
<td>513/1-020</td>
<td>4974</td>
<td><a href="mailto:g.lee@cern.ch">g.lee@cern.ch</a></td>
</tr>
</tbody>
</table>

*CERN PROGRAM LIBRARY OFFICE*

<table>
<thead>
<tr>
<th>Service</th>
<th>Name</th>
<th>Address</th>
<th>Telephone</th>
<th>E-mail Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>All aspects</td>
<td>Jamie Shiers</td>
<td>513/1-002</td>
<td>4928</td>
<td><a href="mailto:cernlib@cern.ch">cernlib@cern.ch</a></td>
</tr>
<tr>
<td>VMS, VM/CMS</td>
<td>Jamie Shiers</td>
<td>513/1-002</td>
<td>4928</td>
<td><a href="mailto:cernlib@cern.ch">cernlib@cern.ch</a></td>
</tr>
<tr>
<td>Apollo, HP/UX Support, SHIFT, CSF</td>
<td>Gunter Folger</td>
<td>513/1-003</td>
<td>5010</td>
<td><a href="mailto:cernlib@cern.ch">cernlib@cern.ch</a></td>
</tr>
<tr>
<td>DECentre, Sun Support</td>
<td>Ian Mclaren</td>
<td>513/1-003</td>
<td>5010</td>
<td><a href="mailto:cernlib@cern.ch">cernlib@cern.ch</a></td>
</tr>
<tr>
<td>Distribution of CERNlib material</td>
<td>Lysiane Besson</td>
<td>513/1-014</td>
<td>4951</td>
<td><a href="mailto:cernlib@cern.ch">cernlib@cern.ch</a></td>
</tr>
</tbody>
</table>

OTHER SERVICES

<table>
<thead>
<tr>
<th>Service</th>
<th>Name</th>
<th>Address</th>
<th>Telephone</th>
<th>E-mail Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop Computing</td>
<td>Chris Jones</td>
<td>31/2-020</td>
<td>4884</td>
<td><a href="mailto:chris.jones@cern.ch">chris.jones@cern.ch</a></td>
</tr>
<tr>
<td>Operating Systems</td>
<td>Les Robertson</td>
<td>31/3-007</td>
<td>4916</td>
<td><a href="mailto:les.robertson@cern.ch">les.robertson@cern.ch</a></td>
</tr>
<tr>
<td>Tape operations</td>
<td>R.P. Minchin</td>
<td>513/R-009</td>
<td>13+5559</td>
<td><a href="mailto:ric.minchin@cern.ch">ric.minchin@cern.ch</a></td>
</tr>
<tr>
<td>Tape purchase</td>
<td>Mario Vergari</td>
<td>513/R-009</td>
<td>13+5602</td>
<td><a href="mailto:m.vergari@cern.ch">m.vergari@cern.ch</a></td>
</tr>
<tr>
<td>Experimental Tape Allocation</td>
<td>Hansjorg Klein</td>
<td>13/3-024</td>
<td>2124,2060</td>
<td><a href="mailto:hans.klein@cern.ch">hans.klein@cern.ch</a></td>
</tr>
<tr>
<td>Computer Science Library, mornings only</td>
<td>Jutta Megies</td>
<td>513/1-024</td>
<td>2379</td>
<td><a href="mailto:j.megies@cern.ch">j.megies@cern.ch</a></td>
</tr>
<tr>
<td>Oracle</td>
<td>Sergio Santiago</td>
<td>31/3-011</td>
<td>4134,13+5580</td>
<td><a href="mailto:s.santiago@cern.ch">s.santiago@cern.ch</a></td>
</tr>
<tr>
<td>Computer Security</td>
<td>John Gamble</td>
<td>31/3-030</td>
<td>3105</td>
<td><a href="mailto:j.gamble@cern.ch">j.gamble@cern.ch</a></td>
</tr>
<tr>
<td>Central Vax Service Manager</td>
<td>Julian Bunn</td>
<td>31/1-025</td>
<td>5029,13+5551</td>
<td><a href="mailto:j.bunn@cern.ch">j.bunn@cern.ch</a></td>
</tr>
</tbody>
</table>

ONLINE COMPUTING: See “ONLINE”, the Newsletter of Data acquisition and Computing for Experiments,

Available from Anne Perrelle 31/1-003 2406 a.perrelle@cern.ch

COMPUTING FOR ENGINEERING: See the CERN Computing Support for Engineering Newsletter,

Available from Monique Tute-Lavergne 513/2-010 2863 m.tate@cern.ch

COMPUTER TIME ALLOCATION GROUP (COCOTIME):

Secretary: A.E.Ball/ECP 14/6-023 3849 alan.ball@cern.ch

DIVISIONAL REPRESENTATIVES FOR COMPUTING

<table>
<thead>
<tr>
<th>Div.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS</td>
</tr>
<tr>
<td>ECP</td>
</tr>
<tr>
<td>PPE</td>
</tr>
<tr>
<td>TH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Div.</th>
<th>Representative</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS</td>
<td>M. Draper</td>
</tr>
<tr>
<td>ECP</td>
<td>A. Grant</td>
</tr>
<tr>
<td>PPE</td>
<td>H. Klein</td>
</tr>
<tr>
<td>TH</td>
<td>R. Sommer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Div.</th>
<th>Representative</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>P. Heymans</td>
</tr>
<tr>
<td>FI</td>
<td>S. Lauper</td>
</tr>
<tr>
<td>PS</td>
<td>J. Riche</td>
</tr>
<tr>
<td>TIS</td>
<td>G.R. Stevenson</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Div.</th>
<th>Representative</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td>A. Koppanyi</td>
</tr>
<tr>
<td>MT</td>
<td>C. Delamare</td>
</tr>
<tr>
<td>SL</td>
<td>A. Hilaire</td>
</tr>
<tr>
<td>ST</td>
<td>M. Draper</td>
</tr>
<tr>
<td>PE</td>
<td>D. Duret</td>
</tr>
<tr>
<td>ST</td>
<td>G. Martin</td>
</tr>
</tbody>
</table>

DIVISIONAL REPRESENTATIVES FOR NETWORKING

<table>
<thead>
<tr>
<th>Div.</th>
<th>Representative</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS</td>
<td>M. Moller</td>
</tr>
<tr>
<td>FI</td>
<td>J. Groz</td>
</tr>
<tr>
<td>PS</td>
<td>A. Pace</td>
</tr>
<tr>
<td>TIS</td>
<td>B. Moy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Div.</th>
<th>Representative</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>R. Saban</td>
</tr>
<tr>
<td>MT</td>
<td>C. Andrews</td>
</tr>
<tr>
<td>SL</td>
<td>P. Lienard</td>
</tr>
<tr>
<td>TIS</td>
<td>L. P. Stevenson</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Div.</th>
<th>Representative</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSU</td>
<td>M. Moller</td>
</tr>
<tr>
<td>PE</td>
<td>D. Duret</td>
</tr>
<tr>
<td>ST</td>
<td>G. Martin</td>
</tr>
<tr>
<td>ECP</td>
<td>J.-P. Porte</td>
</tr>
<tr>
<td>PPE</td>
<td>E. Pagiola</td>
</tr>
<tr>
<td>TH</td>
<td>R. Sommer</td>
</tr>
</tbody>
</table>
UNIX Workstations and X Terminals Support

List of Contact Accounts

Team Leader          Alan Silverman                      31-1-030 tel 4955
                    e-mail - Workstation.support@cern.ch
UNIX Workstations and X Terminals Front Desk
Christiane Ball       31-1-015 tel 3349
                    e-mail - Workstation.frontdesk@cern.ch
SUN software          e-mail - sun.support@cern.ch
DEC's OSF/1 s/w       e-mail - osf.support@cern.ch
RS/6000 s/w           e-mail - aix.support@cern.ch
ULTRIX s/w            e-mail - ultrix.support@cern.ch
HP 700 s/w            e-mail - hp.support@cern.ch
Apollo/Domain         e-mail - apollo.support@cern.ch
X terminals           e-mail - xterminal.support@cern.ch
SGI software          e-mail - sgi.support@cern.ch
AFS                   e-mail - afs.support@cern.ch
Printers              e-mail - printer.support@cern.ch

COMMUNICATIONS AND NETWORKS

Please use generic electronic mail addresses whenever possible. These mailboxes will be read even when the usual specialist is absent. It is helpful to use relevant keywords in the subject field of your message. Contact specialists directly only for very urgent cases or for very general consultations.

Service
Software and interface consultancy
Network infrastructure consultancy (backbone and FDDI)
Network infrastructure consultancy (Ethernet in buildings)
Network security alerts (hacking attacks)

General network operational problems

Specific Ethernet problems
Internet (TCP/IP) registration requests
LAT terminal server registration requests
DECNet Registration Requests and Queries
Other registration requests
Queries about electronic mail
Novell coordination and general PC networking advice
Advice on Macintosh networking

E-mail Address
John.Garble@cern.ch or Mike.Gerard@cern.ch
Joop.Joosten@cern.ch or Jacques.Rochez@cern.ch
Leo.Sohe@cern.ch
cert@cern.ch (if urgent: 4927 or 8665)
[5011 night/weekend]
netops@cern.ch (if urgent: 4927 or 8665)
[5011 night/weekend]
ether-support@cern.ch or 2299
tcppip@cern.ch
lat-support@cern.ch (2299 for problems)
decnet-support@cern.ch (dxmibt:decnet-support)
etops@cern.ch
mail-support@cern.ch
nice@cern.ch
macnet@cern.ch (2299 for problems)
[4366 for general Macintosh problems]

SEMINARS AND PRESENTATIONS

<table>
<thead>
<tr>
<th>Designation</th>
<th>Organizer</th>
<th>Div.</th>
<th>Address</th>
<th>Telephone</th>
<th>E-mail Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing Colloquia (Auditorium / bldg 500)</td>
<td>Frederick James</td>
<td>CN</td>
<td>513/2-012</td>
<td>4985</td>
<td><a href="mailto:f.james@cern.ch">f.james@cern.ch</a></td>
</tr>
<tr>
<td>Computing Seminars</td>
<td>Douglas Kemp</td>
<td>CN</td>
<td>31/1-011</td>
<td>5024</td>
<td><a href="mailto:d.kemp@cern.ch">d.kemp@cern.ch</a></td>
</tr>
<tr>
<td>(CN Auditorium 31/3-005)</td>
<td>Jean-Pierre Porte</td>
<td>ECP</td>
<td>32/2-C07</td>
<td>3457</td>
<td><a href="mailto:j.pierre.porte@cern.ch">j.pierre.porte@cern.ch</a></td>
</tr>
<tr>
<td>Technical Presentations</td>
<td>Richard Keyser</td>
<td>SL</td>
<td>864/1-B23</td>
<td>4363</td>
<td><a href="mailto:r.keyser@cern.ch">r.keyser@cern.ch</a></td>
</tr>
<tr>
<td>(CN Auditorium 31/3-005)</td>
<td>Stanley Cannon</td>
<td>CN</td>
<td>513/S-014</td>
<td>5036</td>
<td><a href="mailto:stan.cannon@cern.ch">stan.cannon@cern.ch</a></td>
</tr>
</tbody>
</table>
1. General

1.1 Letter from the Editor

Nicole Cremel CN/ASD

The objective of this article is to make a review of what are the main purposes of a “Computer Newsletter” at CERN, and of how could it be improved. It is also the result of a short discussion we have had on this subject inside the CN division. I would be pleased now to have the feedback of the CNL readers: general comments or comments on what follows are welcome (to be sent directly to me: cnl.editor@cern.ch).

Concerning the general “look” of the CNL one should be aware that we have to make a compromise between the purely esthetic side and the cost and time to produce it, but also the possibility for external sites to print it. I think that the ability to keep the whole CNL as a PostScript file is appreciated. Anyway, any idea which would make the CNL more attractive and “user-friendly” to you, and would not be too difficult to implement will, be welcomed.

However, the most important point is certainly the contents. For some time now, we have tried to have some chapters on a regular basis, e.g.:

- UNIX Workstations and Desktop Support
- Communications and Networks
- Program Library and Application Software
- Tutorial Section
- Text Processing

We think that we probably could extend this (non-exhaustive) list to some new chapters, that have either never appeared, or only rarely, but should do so in the future. For instance:

- VM to UNIX Migration task
- Databases

Most of these chapters could or should be always present, and, in the case that no special contribution has been provided at a given time, they might appear with the sentence “No news items on that subject” (especially for the more important chapters).

The past rule which consisted of publishing a CNL together with a CERN library release is not true anymore. With the increasing number of new hardware platforms to be supported, the number of CERN library releases is expected to decrease to not more than two a year, whereas we think that the CNL should be published about four times a year (before and after the summer holidays, before Christmas and around Easter). Consequently the chapter “Program Library and Application Software” will no longer be the reference for release information. Only the announcement of a future release will be there, with the main features for a given package. For detailed information readers will have to refer to the corresponding news items which will be put out at the time of the release on the various central computers.

I would like to know if the recent Fortran 90 and LaTeX tutorials have been appreciated, and eventually find some volunteers to contribute similar tutorials, e.g. on C, C++, Unix, X-Windows, etc. Possible subjects could also include tools recommended by the UNIX Migration Task Force. Volunteers could be people who already have the written material (on a computer), even in plain text, for a formal or informal course they may have given at CERN or elsewhere. I can do the formatting job if this is not a huge amount of work.

Another chapter could be dedicated to user input. This has been tried several times in the past in the form of a “Vox Populi” section. This was not always successful and this section finally disappeared. Now there is no real way for the reader to comment on ideas, news or articles that have been published in previous CNLs. I propose to try again and include in future editions a “Letters to the Editor” section which will provide this possibility.

Suggestions were made to simplify the scheme used to accept incoming articles and I will accept from now on all articles directly sent to me, the CNL editor (cnl.editor@cern.ch), without particular constraints on the mail subject line. The “CNL machine” on CERNVM (e-mail cnl@cern.ch) will continue to handle automatic mechanisms with specific key words given on the mail subject line (“Subject: …”) such as:

- Subject: HELP to know more about what the CNL server can do for you,
- Subject: ANNOUNCE POSTSCRIPT CNL to get a mail whenever there is a new CNL ready as a PostScript file on the ASIS server.

We now have three types of “Computer Newsletters” at CERN:

- The CSEN ("Computing Support for Engineering NewsLetter"), whose purpose is to keep the engineering community at CERN informed on the availability of support for its particular computing needs and on the direction in which this support is developing. It is more oriented towards the community using Computer Aided Engineering (CAE) tools.
- The MMCNL ("CERN Mini and Microcomputer NewsLetter"), published about four times a year, contains reports on developments, and up-to-date information on services, especially oriented towards users with an interest in data-acquisition and related problems.
- and the CNL ("CERN Computer NewsLetter"), which is more general than the two others.
It may happen that the same article has to be submitted to all three. In that case, the best and easiest way is for the authors to send an ASCII file, as the three Newsletters do not use the same text preparation tool.

Finally, I would like to say that I would very much appreciate earlier submission of articles. For improving that, I suggest that people issuing any kind of news or publication should think at the same time about the usefulness of publishing it subsequently in the CNL (maybe in a shorter or extended form) and immediately send it to me: cnl.editor@cern.ch (not waiting for the deadline, because there is a good chance those people will forget to do it by that time). I am sure the editorial board can count on the cooperation of our readers concerning all these matters.

1.2 User (Account) Registration

Harry Renshall CN/PDP

(Physics Services Coordinator)

Users who wish to belong to multiple computer groups, usually because they are working on several experiments, would normally prefer to use a single userid (login name). Currently this feature, multiple group (or for Unix systems multiple gid) association with the same userid, is allowed in the USER-REG scheme for account registration only for AFS, WorkGroup services (CUTE services) and CERNVM. CORE services (CSF, SHIFT services) are unable to support this feature.

Whilst we fully recognize the need to support this feature in CORE and are working to implement it, we have decided that until such time, all “modern” services (CORE, PARC, CUTE) will be restricted to a single gid (and for tape access a single uuu) for a given userid. In other words we believe across-service consistency is the more important issue. Such a consistency will enable us to provide tape access in CORE for all users of CUTE and PARC services. As most of you are already aware, CORE tape (and staging) access is becoming the mainline central way of accessing tape data.

The user/account registration interface, USERREG, will shortly be modified to allow only one group code to be associated with a Unix login identifier across the CORE, PARC and CUTE services. We hope this will be a temporary restriction. Any users whose accounts will have to be modified to comply with the single group (gid) and uuu association to a given userid will be contacted individually.

In order to allow AFS file sharing from different userids, the AFS registration will not be changed; a given userid may still be associated to multiple groups (gids).

1.3 ASIS Anonymous ftp Server

Philippe Defert CN/DCI

For our ease of maintenance, and for your comfort, please use from now on the name ‘asisftp.cern.ch’ for the ASIS anonymous ftp server. In the near future the name asis01 will no longer be recognized as an ftp server.

1.4 Optimizing Fortran compiler available for Solaris 2.x—free

Irene Lequerica CN/GPM, Mike Metcalf CN/ASD

The GPMIMD-2 project has just taken delivery of a 32-node Meiko CS-2. As the result of various technical and financial considerations, the project is acquiring the Edinburgh Portable Compilers (EPC) Fortran 90 compiler for use on each of the nodes. This compiler is fully optimizing and accepts all the usual Fortran 77 extensions. It has thus been tested on the CERN Program Library and with the CERN benchmark. The terms of the contract are such that we can offer, free of charge, use of this compiler on any Solaris 2.x system on the CERN site. Anyone wishing to take advantage of this offer should, in the first instance, contact Irene Lequerica (ireleq@desrv01.cern.ch).
2. VM to UNIX Migration Task

2.1 The Proposed SP2 Central Computer

Harry Renshall CN/PDP

Base Hardware

A proposal to replace the IBM ES9000, which provides the CERNVM service, by a half-sized machine and acquire a 128-node IBM SP2 system will be considered by the CERN Finance committee at the end of September. If approved, the ES9000 will be changed during the 1994 end-of-year shutdown and the SP2 will be delivered in two phases, namely 64 nodes during the third week of October 1994 and the remaining 64 at the end of March 1995. In the current plan all the nodes will be of the “thin” type with 66.7 MHz Power2 processors, the same as in the IBM RS6000/390 workstation, and they will be interconnected by a high-speed switch which has a node-to-node capacity of 40 MBytes/second. A single RS6000/390 gives CERN benchmark results of 25.5 CERN Units at normal optimization and 27.7 at maximum optimization.

The hardware that will be delivered in October is all the nine storage frames, 16 nodes equipped with 128 MB of memory and a 2 GB internal disk and 48 nodes equipped with 64 MB of memory and a 1 GB disk. Eight of the nodes will be equipped with fibre-optic network interfaces and eight of them will be equipped with SCSI interfaces for access to external disks. Already on order is a DEC Storage works cabinet and 60 disks of 2.1 GB formatted capacity intended for direct connection to the SP2. An associated order for DEC Storage works standalone file servers of 200 GB capacity that will be used for public staging space, and hence available to the SP2 service, has also been made.

Overview of Applications and Services

The applications for this system foreseen at present fall into four main categories:

- those related to the replacement of CERNVM
- those related to data services
- the parallel applications for current experiments
- projects and applications of a more experimental nature but of long-term interest for LHC.

These services will be implemented by dedicating one quarter of the machine to serial interactive work, one quarter to serial batch work, some nodes for service functions and the remainder for parallel applications. Operating system software supports such partitioning in a flexible way, and it is foreseen that there will be demand-driven changing of node function such as using some interactive nodes for other work at weekends.

The CERN-wide AFS (Andrew file system) will provide home directory services and access to group level files for all registered users. The current CORE (Centrally Operated Risc Environment) home directories will be read/write mounted via NFS auto-mounting unless an experiment explicitly requests this not be done. The CORE file and tape servers will provide access to the non-AFS bulk data of the experiments using CORE and to all CORE tape services via the experiment and public staging pools. It is not expected that data production or intensive analysis will be done on the SP2 by those experiments already having CORE capacity, but that it might be used by subgroups wanting RS6000 architecture or doing peripheral analysis, and thus the data rates to CORE disk servers are expected to be modest.

Network Connectivity

Each node has an Ethernet adapter and a switch interface each of which requires an IP address and name. Access for terminal traffic, including X, will be over Ethernet. All data intensive traffic, including APS file access, is to be routed over FDDI or Utranet. The system includes Interactive Session Support (ISS) whereby users telnet to a single name and are returned the IP address of the interactive node to use based on user definable criteria such as least number of users or most free memory. The name for the interactive service, which will be the only one visible to full public use, will be CERNSP PLUS. CERN.CH (CERNSP PLUS will be enough from inside CERN). There will be full external world-wide access to the interactive nodes.

System Environment

Each node will run the operating system AIX/6000 V3 R2.5 which will be preloaded onto each node’s local disk as will be the Parallel System Support Programs. Local extensions to AIX will be applied using CERN’s Shrink wrapped Unix Environment utilities. The PSSP allows a single point of intervention, a separate workstation, called the control workstation, for many system management functions such as propagating common files and configuration options to all nodes and collecting accounting data. The license for AIX permits any number of simultaneous users. The system will include a performance toolbox for centralized monitoring of CPU loads and other parameters.

The system will be delivered with the IBM Parallel Operating Environment which includes user-callable parallel application programming interfaces such as PVM and an IBM message passing interface which will converge to a newly-emerging MPI standard. Also included will be a parallel debugger and parallel performance visualization tool.

An ANSI standard C compiler is part of the AIX base and a ten simultaneous user license for the XL C++ compiler is part of the contract. The contract includes a 100-simultaneous user license for the XL Fortran 3.1 Fortran 90 compiler (this
includes the Fortran 77 standard as a subset). The license is
controlled by a license server and applies to all RS6000 nodes
under its control. Batch jobs finding a license unavailable will
wait.

The IBM LoadLEVELler product will be available for batch
work. Clients for job submission from other unix systems
will be available.

All nodes will be AFS clients. Interactive logins will be
trapped by the AFS mechanism and verified by AFS authenti-
cation. A mechanism will be put in place allowing batch jobs
running on serial batch nodes to obtain, and refresh, AFS to-
kens on behalf of the user submitting the job.

The local disks will be used to support AFS file caching, Unix
system and work space requirements and paging.

User Environment

There will be a single separate user registry for the SP2 ser-
vice. Registration will be through the Computer Centre Data
Base. User home directories will be in AFS and the normal
AFS login verification will be performed. The service will
hence follow the common developments in the AFS environ-
ment made for all the public and private work group servers.
Thus user-level disk quota management within group budgets
will become available as will mechanisms allowing batch job
access to user and group AFS files.

The standard DCI group (Desktop Computing Infrastructure)
environment, which is packaged as SUE, the shrink wrapped
unix environment, and includes the Hepix sponsored Unix
shell environments, will be fully supported. The default login
shell for the service, that is obtained when no group or user
preference is given on registration, will be tcsh. The products
and tools defined at their various support levels by the User
Migration Task Force, led by M.Marquina, will be fully avail-
able. This includes special products to encourage migration
off CERNVM such as the REGINA Rexx interpreter and an
Xedit emulator called TIE. The UMTF product set includes
support for ASCII terminals but the emphasis in this environ-
ment is towards usage of bit-mapped screens and the SP2 ser-
vice will follow this emphasis. Note that the CERN Program
Library will be the common RS6000 version.

Batch work, including parallel, will be run under control of
the IBM LoadLEVELller product. It is anticipated that work
currently performed on VM as express, short and medium
jobs will mostly be done interactively. The elongation fac-
tors on the SP2, the ratio of real time to CPU time, should
be significantly better than on CERNVM enabling far more
work to be done interactively and longer batch work to be
run. The number of interactive nodes has been defined as
32, anticipating up to 30 simultaneous users per node. The
detailed definition of batch resource limits and class distri-
butions has not been made but the starting point will be 2
to 3 batch jobs running per node with time limits like those
currently on CERNVM but in native time rather than CERN
units time. User job time limits will be in native units as for
all CORE services and converted to CERN units for account-
ing.

All tape access, reading and writing, will be via the CORE
staging mechanism. Access to CORE experiment data files
will be via the Remote File I/O mechanisms of CORE made
available transparently through this code being integrated
into the CERN Program Library.

Supercomputer Aspects

The SP2 is classed by the US government as a supercomputer
when more than eight nodes are used in parallel. The machine
must be in a secure area namely our current machine room.
The machine will be delivered with software which restricts
normal access to less than eight nodes in parallel and there
will be no restrictions on users registering for such normal us-
age. Users requiring more than eight nodes in parallel will be
required to be registered supercomputer users with the current
restrictions. There is a very small number of countries where
such access will be refused and a larger number where appli-
cations will have to be examined individually by the US gov-
ernment (up to now none has ever been refused). There are no
restrictions on any CERN member state national. There will
be no restrictions on world-wide network access to the SP2
for non-supercomputer usage. Controls will be put in place
such that network access from restricted countries will not be
able to operate in supercomputer mode i.e. with more than
eight nodes in parallel.

Service Startup Timescales

The base hardware has been scheduled with IBM manufac-
turing. The Finance Committee will decide on 28 Septem-
ber whether or not to approve the project in its present form.
The basic CERN benchmark has been run and a set of parallel
benchmarks is being prepared. It is intended to run these as
soon as possible in order to verify that the basic parallel ar-
chitecture does not contain any hidden incompatibilities with
CERN applications. At the same time a validation suite of
programs intended to stress aspects of the SP2 system is be-
ing prepared. The system is scheduled for delivery around 20
October and will take up to a week to assemble and configure.
As soon as possible the validation tests will be run for a suffi-
cient length of time that we can verify the correct functioning
of the machine.

We will not recommend general users to obtain SP2 accounts
during 1994. The date for such a recommendation will de-
pend not only on the hardware and software configuring of
the SP2 but on the maturity of the user environment which
includes aspects such as the availability of user training and
documentation. During 1994 we will invite selected users
and will allow also any user who requests an account to have
one. This will be clearly stated to be an experimental service
where we want to know of any problems but would not give
any time scale for fixing them. The preference for migrating
users off CERNVM to the SP2 would be for a steady flow dur-
ing 1995 rather than in a sudden flood, as the user support ef-
fort is expected to be large. By the end of 1995 we should be
left on CERNVM with those users who have applications that
are difficult to move and who may need special help.
3. UNIX Workstations and Desktop Support

3.1 The Zephyr Message Service for Unix Users

Tony Cass CN/DCI and Dan Pop PPE/LE

Zephyr, a message sending system for Unix users, can now be used at CERN. Zephyr has the advantage over commands such as write that you do not need to know where someone is working to send them a message. If your friend is logged into Zephyr then a simple

```
  zwrite my_friend -m Hello
```

will put a window on their X display (or some lines on their ASCII terminal) with your message. Zephyr can be thought of as a Unix equivalent of CERNVM’s TELL or MESSAGE commands. Because of this Zephyr is likely to be used by the computer operators for sending urgent messages—so we recommend you login to the Zephyr system even if you don’t send Zephyr messages yourself.

Zephyr runs on all supported Unix workstations at CERN and the files are available from ASIS. If your workstation is an AFS client you should already have all the necessary code and directories installed. Full details of how to install and use Zephyr are in the “Zephyr at CERN” manual, CN/DCI/165, available from the UCO—or on the Web as http://wwwcn/writeups/zephyr. Simple instructions on installing and using Zephyr are given below.

Besides sending messages between individual users, Zephyr also allows messages to be sent to a group of users. This could be useful to send a message to all members of a collaboration, for example. Again, details of how to use Zephyr in this way are documented in “Zephyr at CERN”.

If you have problems using Zephyr please contact the UCO (e-mail User.Support@cern.ch, phone 4952) or send a mail to Zephyr.Support@cern.ch.

How to Use Zephyr

- Check the code is installed on your workstation. You should have the files zhm, zwgc, zwrite and zlocate in /usr/local/bin and a directory /usr/local/lib/zephyr. If you do not, then see the “Zephyr at CERN” manual for instructions on how to copy these files from ASIS.

- Check your workstation has the following two lines in /etc/services:

```
zephyr-clt 2103/udp # Zephyr Client
zephyr-hm 2104/udp # Zephyr host manager
```

- Each workstation needs one “Zephyr Host Manager” zhm daemon running. If it does not—look for a zhm process with ps -ef | grep zhm (or ps aux | grep zhm for SunOS and Ulitrix systems)—then start one with the command

```
  /usr/local/bin/zhm \
  > 'cat /usr/local/lib/zephyr/server.list'
```

(the quotes are back quotes). This process does not need to run as root.

- To log in to zephyr use the command

```
  /usr/local/bin/zwgc
```

(You do not need an &) You will probably want to include this command in your initialization files—see the “Zephyr at CERN” manual for details.

- To send a zephyr message, use the /usr/local/bin/zwrite command. For example:

```
  > zwrite dougal
  Type your message now.
  End with control-D
  or a dot on a line by itself.
  Hello There!
  dougal: Message sent
  >
```

You can send a short message on one line:

```
  > zwrite dougal -m Hello
```

- To see if someone is logged in to zephyr you can use the /usr/local/bin/zlocate command:

```
  > zlocate tnt
  ues1.cern.ch :0.0 Tue Jul 12 15:13:04 1994
  > zlocate natalie
  Hidden or not logged-in
```
4. Communications and Networks

4.1 The Recabling of CERN

The CN/CS Internal Network Team

The existing general purpose network at CERN consists of an FDDI backbone infrastructure which interconnects a number of Local Area Networks of different types:

1. Ethernet is the main LAN and is used for workstations, servers and PCs. In most buildings Ethernet runs over 10Base2 thin coaxial cables. The following protocols are mainly, but not exclusively, used over this medium:
   - Internet (TCP/IP),
   - DECnet (Phase IV and Phase V),
   - Other DEC proprietary protocols (LAT, MOP etc.),
   - Novell Netware (IPX),
   - AppleTalk.

2. In addition to its backbone role, FDDI is used for connecting high-performance workstations for high energy physics data analysis, mainly using Internet protocols.

3. Apple equipment (Macintosh personal computers and Apple printers), is today mainly connected to LocalTalk or TurboNet cabling rather than Ethernet.

4. ASCII terminals are connected via RS232C to terminal servers attached to Ethernet.

Each type of connection has its own cabling system.

The number of user devices connected to the network is growing regularly and is conservatively estimated as follows:

- more than 6500 Ethernet (of which >4000 in office buildings),
- more than 2000 LocalTalk/TurboNet,
- about 3000 ASCII RS232C terminals.

This situation must be changed for a number of reasons:

- reliability,
- flexibility,
- manageability,
- scalability of size and performance.

To cope with future conditions, the existing general-purpose network must be changed to a more manageable system, installed by industry and with proper industrial support.

Network infrastructure

The first step in this direction will be to install a new network infrastructure with:

- The installation in buildings of a structured cabling system, with radial cables from various star points. Unscreened Unshielded Twisted Pair Category 5 cable (UTP Cat. 5), suitable for all the types of LAN listed above, will be used. The main implementation of the structured cabling system is expected to take three years, from 1995 until 1997, according to current budget plans.
- The extension of the existing FDDI backbone to interconnect all star points of the structured cabling system.
- The exploitation of an industry standard cable management system and database.

The project has taken into consideration all buildings with more than 30 rooms and the smaller buildings immediately adjacent to them. In total, 61 such buildings, requiring up to 18,000 network sockets, have been identified. The remaining rooms are located in small temporary barracks or in buildings mostly used as workshop floors, or experimental halls. For such places, structured cabling will be considered later.

The cabling project is a collaboration between the CN/CS and ST/NC groups, and a contract will shortly be awarded to industry. The precise planning will be arranged with Divisional representatives.

Networking devices

There are two types of networking devices which will be installed. All these devices will be managed by an industry standard network management package. This package will be intimately associated with the cable management package mentioned above.

Networking devices for structured cabling

The network outlets provided by the structured cabling of the buildings will all be identical. Different types of network equipment ("hubs") will be installed in the star points for Ethernet, LocalTalk, etc. It is only when a network outlet is patched through to a hub that it will be usable.

The network equipment which will be installed in the star points are:

- 10baseT Ethernet intelligent multiport repeaters.
- TP PMD (copper FDDI) wiring concentrators.
- LocalTalk/TurboNet concentrators.
- Terminal servers.

Bridges and routers

The general purpose network cannot survive the growing daily problems of software interference and unmanageable traffic growth on a jumbo bridged network as it is today. The time has come to move to a routed network while making the best effort to maintain performance. The key to performance...
in a routed network is locality of traffic and this should be implemented whenever possible. Our ambitious goal is to reduce the bridged network in the office buildings from about 4000 today to fewer than 1500 devices by the end of 1996. The implementation of structured cabling of buildings will offer the opportunity to install in the star points:

- a routed network for IP, IPX (Novell) and AppleTalk systems;
- a residual bridged network for systems running non-routable protocols (e.g. LAT terminal servers and LAT hosts);
- special solutions for high performance requirements.

**Impact on users**

We hope that the main impact on users will be a more reliable and more flexible network with potential for growth. However, when a particular building is re-cabled there will be some immediate impacts:

- Unavoidable disturbance caused by the work to be done. Every effort will be made to minimize this and give as much warning as possible.
- An important change in working habits will occur. In a building with structured cabling, all requests to connect new equipment to the network, or to move existing equipment, must be handled in a systematic way. The procedure for this will be announced in due course.
- As routers are introduced, workstations etc. using TCP/IP will have to be allocated a new IP address. For Novell and AppleTalk, renumbering will be automatic.

### 4.2 CERN Networks Status Available via WWW

**Mike Gerard CN/CS**

There is now the possibility to see the current status of the CERN general purpose ethernet via any WWW browser (normally Mosaic). The relevant URL is

http://vscoms.cern.ch:8000/

The information available includes access to the "Trouble Ticket" system used to track all reported network incidents. We always welcome any extra information or comments on any of these reports.

Also available is a list of blocked ethernet addresses. These are the physical addresses of systems which have caused problems such that they have been automatically or manually blocked in all of our ethernet to FDDI bridges, so that they cannot have any access other than very local. We always try to inform the owner of the equipment about such blockages, but often the owner is unknown, has left CERN or is unavailable by phone or e-mail. Therefore, if you have some equipment which appears to have problems communicating over the network it is worthwhile to check whether its address is in the blocked list before calling to say that the network is faulty.

Please note that some of the items available require that the WWW browser be able to handle forms. The Macintosh Mosaic version currently in use does not have this ability: it will be included in version 2 (currently in alpha test).

### 4.3 Running DECnet/OSI (Phase V) at CERN

**Denise Heagerty CN/CS (CERN DECnet Coordinator)**

If you are responsible for VMS systems at CERN, it is time to be planning your installation of DECnet/OSI (Phase V).

DEC announced sometime ago that the Phase IV-only product, which most CERN VMS systems are running, will not be supported beyond VMS 6.1. We highly recommend that you start to get experience with DECnet/OSI once you have upgraded some systems to VMS 6.1. This will give you feedback to plan DECnet/OSI installations for all your VMS systems.

DECnet/OSI is installed as a layered product. Whilst the software needs only to be installed once per system disk, a configuration of DECnet/OSI needs to be made on each node. The configuration program includes tools to convert existing Phase IV data bases. From a user perspective, the move to DECnet/OSI should be transparent (commands and programs should run as before). The main change is in the way that DECnet is started and managed (NCP commands and data bases are replaced by NCL commands and a distributed name service).

The DECnet/OSI software is being continually improved by DEC, mainly in the area of ease of installation and configuration. To be sure you use the latest version, please contact dxmint::decnet-support just before you plan to install it. At that time, we will also supply you with the help you need to configure DECnet/OSI for the CERN environment.

We are in the process of organising a 1-day DECnet/OSI seminar for VMS system managers (provisionally planned for 11 Nov 1994). Further information about this will appear in the VAX Managers’ News Group.
5. Text Processing

5.1 Text Processing: An Update

Michel Goossens CN/ASD

\LaTeXe\ developments

As announced in previous CNL's the standard release of \LaTeXe\, known under the name \LaTeXe\, is now publicly available on the CTAN archives, and is installed on the Unix, VMS and VM/CMS systems. I can be accessed by typing \texttt{latex2e(12e on VMS)} instead of \texttt{latex}.

\LaTeXe\ introduces many additions and improvements (see my article in CNL216) with respect to \LaTeXe\ 2.09. It should be noted that files, marked up for standard \LaTeXe\ 2.09 can be run without modifications through \LaTeXe\ 2e in so-called compatibility mode. However, if you want to use the new features of \LaTeXe\ 2e, you must make the necessary few changes.

I anticipate changing to \LaTeXe\ 2e as the standard \LaTeX\ system in the middle of January 1995.

Completing the Alpha VMS \TeX\ system

Thanks to Vassiliy Malishev, a visitor of IHEP (Protvino, Russia), who is acting as a consultant on text processing for the next few months, most utilities previously available on VAX VMS have now been ported to Alpha VMS, including xvi, dvit2tty, dvitovdu, and spell, and many of the programs that are part of the "basic" \TeX\ setup.

5.2 "xwpick" New Version of "xpick" (Pick Images on an X11-screen)

Evgenii Tcherniaev, IHEP Protvino, Russia

The program \texttt{xwpick} (the previous version called \texttt{xpick} was described in CNL 214) lets you pick an image from an arbitrary window or rectangular area of an X Windows (DEC Windows) server and write it to a file in a variety of formats: different PostScripts, GIF, PCX (IBM PC), PICT (Macintosh), PPM (PBM Plus). The name of the program has been changed to avoid a conflict with \texttt{xpick} by Gerry Tomlinson.

The main feature of \texttt{xwpick} is that it produces very compact PostScript files: several times smaller than is possible with other capturing tools, like xgrabsc, xv, xtoPS.

New features

The \texttt{xpick} program sometimes produced faulty PostScript files for large windows with a lot of space of the same colour. This bug has been fixed.

The list of options has been increased to four, and a few further options are already planned for the next release. The current list of options is as follows:

- \texttt{-local} When \texttt{-local} is in the parameter list, \texttt{xwpick} picks an image from the window under the mouse pointer. This option is intended to pick images from pop-up menus which are on the screen only when a mouse button is pressed and disappear immediately after the button is released.

- \texttt{-window id} Pick an image from the window with integer identifier equal to \texttt{id}. To pick the entire screen (root window) the user may use the word \texttt{root} as an identifier. The \texttt{id} number for a window can be obtained with the X Window utility \texttt{xwininfo}.

- \texttt{-gray} Transfer image to gray scale. This option can be used to optimize output on level 1 PostScript printers. On level 2 PostScript printers there is practically no difference in rendering time between colour and gray scale images.

- \texttt{-reverse} Transfer image to reverse colours. It can be used to save toner on gray scale printers when the image has a lot of dark space.

On VAX/Alpha OpenVMS a \texttt{xwpick} help file is available and one can now use VMS style input for the parameters, e.g. \texttt{xwpick/gray/window=0x8400 pict.gif}

What will be new in next release?

The next version 2.20 of \texttt{xwpick} will be released in the near future. Some changes will be introduced in the default behaviour of the program and in the list of options.

In the current version of \texttt{xwpick} the <\texttt{SPACE}> bar interaction is obligatory, i.e. after the rectangular area has been selected the user should press the <\texttt{SPACE}> bar to pick the selected image. In \texttt{xwpick} 2.20 this interaction will not be invoked unless the \texttt{-pause} option is specified.

At least two new options will be added:

- \texttt{-pause} this option will invoke the <\texttt{SPACE}> bar interaction to give the user a possibility to change the selected area just before outputting to a file.

- \texttt{-format} \texttt{frm} where \texttt{frm} is one the following strings: \texttt{ps}, \texttt{eps}, \texttt{epsf}, \texttt{gif}, \texttt{pcx}, \texttt{pict}, \texttt{ppm}. This option will allow one to choose the output format without regard to the extension in the file name. When \texttt{-format} is in the parameter list and the output file name is omitted, the output will be directed to the standard output stdout, this allowing \texttt{xwpick} to be used in pipes, e.g.,

\texttt{xwpick -format ps -window root | lpr}

How to obtain \texttt{xwpick}

\texttt{xwpick} is available on the \texttt{asifsftp.cern.ch} file server in the directory \texttt{/pub/X11/contrib/applications:} as two files \texttt{xwpick-2.10.README} and \texttt{xwpick-2.10.tar.gz}.

Please address your comments, suggestions or bug reports to the author \texttt{via e-mail at chernaev@mx.ihep.su.}
6. Tutorial Section

6.1 Interfacing Fortran and C

Alfred Nathaniel CN/ASD

In this article I shall summarize the experience about calling C from Fortran and vice versa.

Over the past two years the originally Fortran-based library package KUIP has been gradually rewritten in C. (KUIP stands for “Kit for a User Interface Package” and provides the user interface for PAW and other applications from the CERN program library. KUIP has been ported to VMS, VM/CMS, MVS, MS-DOS, Windows/NT, and a large number of Unix flavours.)

The rewrite was not a pure “exercise of style” but it appeared to be the best choice in order to improve the quality of the package. Key elements for a user interface package such as dynamic string handling, recursivity, and function pointers are inherent to the C language. Using C allowed the removal of most of the limitations with respect to maximum string lengths built into the original Fortran code. It also facilitated the interface to Motif which is only accessible through C libraries.

The Fortran routines of KUIP have been replaced by C functions. The C code emulates the former Fortran entry points and is therefore completely compatible.

When doing inter-language calls between Fortran and C we have to realize that the Fortran part is fixed, and that the C code has to adapt to the characteristics employed by the Fortran compiler. The cfortran.h header file by Burkhard Burow provides convenient preprocessor macros for interfacing Fortran and C. For reasons explained below it does, however, not cover the IBM mainframe systems. Also, cfortran.h exercises cpp to the full which always involves the risk of stretching the C preprocessor beyond its limits.

By no means do we want to discourage the use of cfortran.h. However, if cfortran.h cannot be used (because the target platform is not supported or the overhead is considered too large), here you should find the necessary information for doing the Fortran-C interface “by hand”.

1- Type Definitions

It is good style to use typedef names to map the Fortran data types to their C counterparts. Up to now all platforms are satisfied by a single set of definitions:

typedef int INTEGER;
typedef INTEGER LOGICAL;
typedef float REAL;
typedef double DBLPREC;
typedef struct { REAL re; REAL im; } COMPLEX;

but, for example, a Fortran compiler with AUTODBL option for generation 64-bit code would require different mappings.

2- External Identifiers Naming

First of all, there are differences in the naming of external identifiers (SUBROUTINE, FUNCTION and COMMON block names in Fortran and their C equivalent).

2.1- SUBROUTINE

For a Fortran CALL SUB the corresponding C routine has to be named:

SUB all upper case on Cray with cft77 compiler
sub all lower case on Apollo with ftn compiler
sub_ case insensitive on IBM/370 and VMS

The HP/UX and RS-6000 Fortran compilers need a special switch (+ppi and -gextname, respectively) in order to generate sub_ instead of sub. CERN program library policy is to use the form with the appended underscore to avoid the risk of name clashes between Fortran user routines and C library functions.

For convenience I decided to define the capitalized form Sub as a preprocessor identifier:

#define Sub F77_NAME(sub,SUB)

where the preprocessor macro F77_NAME maps to the appropriate external name:

/* Cray, VM/CMS, MVS, VMS */
#define F77_NAME(name,NAME) NAME

/* Apollo FTN */
#define F77_NAME(name,NAME) name

/* others */
#define F77_NAME(name,NAME) name#

The IBM C/370 compiler requires in addition a

#pragma linkage(Sub, FORTRAN)

in order to use the correct argument passing mechanism which is different between Fortran and C. Unfortunately the #pragma cannot be included into the F77_NAME macro and must be specified separately.

1 This information will also be published as an appendix in the KUIP manual.
2 cfortran.h and f2h for generating interface definitions from Fortran source code are available as part of the CERN program library distribution.
2.2- FUNCTION

The same rules as for SUBROUTINE apply also for FUNCTION names. We found, however, that only INTEGER and LOGICAL FUNCTIONs can be emulated in C. REAL FUNCTIONs and others are not guaranteed to work on all platforms.

2.3- COMMON

Fortran COMMON blocks can be accessed from C code. Most compilers apply the same rule for generating the external identifier as for routines but up to now we came across two exceptions: the Convex compiler adds an underscore both in front and at the end. The Asoft Fortran compiler for the NeXT uses the plain lower-case name without underscores. Again for convenience we define the capitalized form Com as a preprocessor identifier:

#define Com F77_BLOCK(com,COM)

where the preprocessor macro expands to the appropriate identifier:

/* Convex */
#define F77_BLOCK(name,NAME) __#name__

/* NeXT */
#define F77_BLOCK(name,NAME) name

/* others */
#define F77_BLOCK(name,NAME) \
    F77_NAME(name,NAME)

Naively one would assume that the C structure mapping the COMMON block should be a declaration (i.e. "extern struct ...") since the storage should be allocated by the Fortran side. Some platforms, however, require the form of a definition, i.e. leaving out the extern keyword.

In order to cope with this difference we define

/* Apollo, Convex, Cray */
#define EXTERN

/* others */
#define EXTERN extern

One last complication is that on Apollo the C definition has to be given a special attribute. Otherwise the linker allocates the C structure separate from the COMMON block it should map.

Again we can define a system-dependent preprocessor macro to cope with this difference:

/* Apollo */
#define F77_COMMON(name) name __attribute__((__section(name)))

/* others */
#define F77_COMMON(name) name

Having these ingredients it is straightforward to translate any Fortran COMMON block into a C struct. For example,

    INTEGER I
    REAL X
    DOUBLE PRECISION D
    COMMON /COMX/ I,X(3,3),D

    CHARACTER*80 CTEXT(10)
    COMMON /COMC/ CTEXT

is described by

#define Com F77_BLOCK(com,COMX)

EXTERN struct {
    INTEGER i;
    REAL x[3][3];
    DBLPREC d;
} F77_COMMON(Comx);

#define Com F77_BLOCK(comc,COMC)

EXTERN struct {
    char text[10][80];
} F77_COMMON(Comc);

and the values are accessible as Comx.i, Comx.x[1][2] for X(3,2), Comc.text[4][0] for CTEXT(5)(1:1) etc.

Note that one should not rely on the possibility to extend the COMMON block size by changing the dimensions in the C code. It is platform dependent whether

EXTERN struct {
    char text[1000][80];
} F77_COMMON(Comc);

without changing the Fortran definition accordingly will actually reserve the requested amount of memory for the COMMON block.

Also one has to be careful with the placement of DOUBLE PRECISION variables. They should always be at an even offset from the start of the COMMON block because otherwise in a case like

    REAL X,Y
    DOUBLE PRECISION D
    COMMON /COM/ X,D,Y

it is not guaranteed that it can be mapped as

struct {
    REAL x;
    DBLPREC d;
    REAL y;
} ...
While the C compiler is allowed to add padding in order to align data, the Fortran compiler is obliged to allocate X and D at consecutive machine words. Many CPUs signal a bus error upon unaligned access to DOUBLE PRECISION variables, and compilers usually add protective code and issue a warning message.

One exception is the HP/UX Fortran compiler which silently adds padding in the same way the C compiler does. That makes the Fortran-C interface easier but at the same time violates the rules of Fortran storage association. In principle, the same COMMON block could be declared in a different routine as

\[
\text{REAL } Z \\
\text{COMMON } /\text{COM/ } Z(4)
\]

and for the HP/UX Fortran compiler Y and Z(4) do not refer to the same memory location anymore.

### 2.4- \text{/PAWC/}

Some of the complications in accessing COMMON blocks from C could be avoided by defining a structure pointer which is filled by a call from a Fortran routine passing the first variable in the COMMON block.

In fact in KUIP this possibility is used for the \text{/PAWC/} Zebra store. The C structure definition looks like

```c
#define Pawc kc_pawc

EXTERN struct COMMON_PAWC {
    INTEGER NWPAR;
    INTEGER IXPAWC;
    INTEGER IHBOOK;
    INTEGER IXHIGH;
    INTEGER IKUIUP;
    INTEGER IFENCE[5];
    INTEGER IQDATA[3];
    INTEGER IQDATA[999];
} *Pawc;
```

and the pointer is initialized by a CALL KIPAWC(NWPAR) to

```c
void Kipawc( struct COMMON_PAWC *pawc )
{
    Pawc = pawc;
}
```

Linking to \text{/PAWC/} in this way was introduced due to a request from W. Wojcik of IN2P3 in Lyon. They wanted to exploit the option \text{DC(name)} of the VS-Fortran compiler which allows to allocate COMMON blocks at execution time. The generated code references COMMON block variables indirectly through a pointer to dynamic memory rather than as an absolute address resolved by the loader. (The libraries produced on CERNVM do not make use of the \text{DC} option. The C code works either way without change.)

Preprocessor macros allow to write C code accessing the Zebra store in a Fortran-ish style. For example, the equivalent of

\[
\text{DO 10 I = 1, IQ(LX-1)} \\
Q(LX+1) = 3.14 \\
10 \text{ CONTINUE}
\]

can be written as

```c
#define IQ(n) Pawc->LQDATA[n-1]
#define IQ(n) Pawc->IQDATA[n-1]
#define Q(n) (((REAL*)(Pawc->IQDATA))[n-1])

for( i = 1; i <= IQ(lx-1); i++ ) {
    Q(lx+i) = 3.14;
}
```

### 3- Argument Passing in Calling C from Fortran

#### 3.1- Numeric data

Passing numeric arguments between Fortran and C is straightforward. One only has to remember that Fortran always uses \text{call-by-reference}, i.e. it passes the address of the memory location where the value is stored.

Consequently, the Fortran routine call

\[
\text{CALL SUB(X,2)}
\]

corresponds to the C function declaration:

```c
void Sub( REAL* xp, INTEGER* ip )
```

The value itself is accessed as in

```c
float x = *xp;
```

To store a value back into the Fortran variable X one has to write in C

```c
*xp = 3.14;
```

#### 3.2- Arrays

If X is an array the third element \text{X(3)} (in Fortran) has to be addressed as \text{xp[2]}, since in C array indices start at 0.

It is important to note that the pointers received as arguments must not be used as local variable. For example, one might be tempted to write:

```c
void Sub( REAL* xp, INTEGER* ip )
{
    int i;
    for( i = *ip; i > 0; i-- ) {
        float x = *xp++;
    }
}
```
where xp is incremented to point to the next element, which is perfectly legal C. But there is a potential problem if Sub is called from Fortran.

Some machines (e.g. IBM/370 and Convex) use argument blocks, i.e. the call passes the address of a memory region containing the argument addresses. Since a Fortran routine SUB would have no way the alter the content of the argument block, the Fortran compiler treats it as a compile time constant. As a result, only the first call to Sub will access the correct array elements.

3.3- LOGICAL

The representation of .TRUE. varies between platforms. It seems to be safe to test a LOGICAL input against 0 but output parameters should use the proper .TRUE. and .FALSE. constants. Returning the C values -0 or 0 will work in tests such as

```
IF(L) ...
```

but may fail for

```
IF(L .EQ. .TRUE.) ...
```

3.4- EXTERNAL

C entry points can receive Fortran EXTERNAL arguments as in

```
EXTERNAL XT
CALL SUB(XT)
```

Most platforms pass directly the address of the routine
typedef void SUBROUTINE();
void Sub( SUBROUTINE* xt )
```
but some compilers (AIX/370, IBM/370, Apollo, NeXT) store the routine address in a memory cell whose address the called function receives. If we undo the direction
void Sub( SUBROUTINE** xtp )
```
{  
    SUBROUTINE* xt = *xtp;
    ...
}
```
in the routine header we can use afterwards in both cases the same C statement:
```
(*xt)(...)
```
for calling the routine through the function pointer.
Using a typedef name does more than just improve the readability of the declaration. For the IBM C/370 compiler it is essential to add
```
#pragma linkage(SUBROUTINE,FORTRAN)
```
in order to declare that function pointers of type SUBROUTINE have to use the Fortran convention of argument passing.

3.5- CHARACTER

This is one of the most tricky and tedious part of Fortran/C intermixing. A Fortran CHARACTER variable is characterized by two items of information—the address and the length—and the mechanism for passing these two values is compiler dependent. We can label the different mechanisms according to the number of arguments received by the called routine. For a SUBROUTINE with N parameters in total of which k are of type CHARACTER the following number of arguments can be passed:

```
N          Address and length are passed in a single string descriptor argument.
N+k        The address is passed at the position of the CHARACTER argument and for each CHARACTER argument the length information is added at the end of the regular argument list.
2N         Like the previous case but there is an extra length indicator, for each CALL argument.
```

With variations on the type of the length indicator these are the three different schemes I have come across up to now. Of course, there are more possible permutations a new Fortran compiler may choose from.

To give a more complete picture we will take the example of the following basic Fortran routine

```
SUBROUTINE SUB(X,CH,Y)
REAL X,Y
CHARACTER*(*) CH
```

and give the different headers, according to machines, for the C equivalent function in order to make it Fortran callable without any problem. In each case we will end up in the C function with the fixed variable names char* ch_ptr and int ch_len containing the address and length of CH.

VMS

VMS passes the address of a string descriptor containing the address and length information as structure elements:

```
#include <descrip.h>
```

```
void Sub( REAL* x,
    struct dsc$descriptor_s* ch_dsc,
    REAL* y )
{
    char* ch_ptr = ch_dsc->dsc$s$pointer;
    int ch_len = ch_dsc->dsc$s$w$length;
    ...
}
```

Cray

Cray packs both items into the 64-bit word received for each argument (and the bit fields used are actually processor dependent):
#include <fortran.h>

void Sub( REAL* x, _fcd ch_dsc, REAL* y )
{
    char* ch_ptr = _fcdtoctp(ch_dsc);
    int ch_len = _fcdlen(ch_dsc);
    ...
}

VS-Fortran
IBM VS-Fortran uses the 2N scheme with call-by-reference for the length information:

#pragma linkage(SUB, FORTRAN)

void Sub( REAL* x, char* ch_ptr, REAL* y, int* x_size, int* ch_ref, int* y_size )
{
    int ch_len = *ch_ref;
    ...
}

NeXT
For the Absoft Fortran compiler on NeXT it looks similar except that call-by-value is used:

void Sub( REAL* x, char* ch_ptr, REAL* y, int x_size, int ch_len, int y_size )
{
    ...
}

Others
The bulk of Fortran compilers add the length as int only for type CHARACTER:

void Sub( REAL* x, char* ch_ptr, REAL* y, int ch_len )
{
    ...
}

Apollo
The only remaining exception is the Apollo ftn compiler which passes the length in a short:

void Sub( REAL* x, char* ch_ptr, REAL* y, short* ch_ref )
{
    int ch_len = *ch_ref;
    ...
}

KUIP offers a number of convenience functions (fstrdup, fstrset, ...) to convert between fixed length, blank-filled Fortran CHARACTER buffers and null-terminated, blank-trimmed C strings. For details refer to the KUIP manual.

4- Calling Fortran from C

Calling a Fortran routine from C is basically the reverse operation of providing a C routine which is Fortran callable. One has to know the correct spelling for the external identifier, for numeric variables pass the address instead of the value itself, and for CHARACTER items construct the appropriate argument list.

We continue to use the example of a subroutine with the same calling sequence SUB(X, CH, Y). In the following code fragments we assume that

REAL x, y;
char* ch_ptr;
int ch_len;

are already filled and that we want to pass their values to Fortran. Again the platform dependence in passing character data causes the main difficulty.

VMS
On VMS we have to fill the elements of the string descriptor and then pass its address:

#include <descrip.h>

struct descs$descriptor_s ch_dsc;
ch_dsc.dsc$w_dtype = DSC$K_DTYPE_T;
ch_dsc.dsc$w_class = DSC$K_CLASS_S;
ch_dsc.dsc$w_length = ch_len;
ch_dsc.dsc$w_pointer = ch_ptr;
Sub( &x, &ch_dsc, &y );

Cray
On Cray a predefined macro packs the address and length information into a cft77 descriptor:

#include <fortran.h>

Sub( &x, _cptofcd(ch_ptr, ch_len), &y );

NeXT
Sub( &x, ch_ptr, &y, (sizeof x), ch_len, (sizeof y) );

Apollo ftn
The byte ordering of the Motorola-68k CPU does not allow the passing of the address of a 4-byte int if a 2-byte short is expected. Therefore for the Apollo ftn compiler we have to use an auxiliary short:

short ch_ref = ch_len;
Sub( &x, ch_ptr, &y, &ch_ref );

Others
For all other compilers except IBM-C/370 the call is straightforward:
Sub( &x, ch_ptr, &y, 
    ch_len );

C/370
The IBM C/370 compiler does not allow to call a VS-Fortran routine with CHARACTER arguments. The naive approach

#pragma linkage(SUB, FORTRAN)

int x_size = (sizeof x);
int y_size = (sizeof y);
Sub( &x, ch_ptr, &y, 
    &x_size, &ch_len, &y_size );

do not work. VS-Fortran passes the argument count implicitly by setting the sign bit of the last word in the argument block. uses argument blocks containing the reference addresses.

For routines with CHARACTER arguments both the words at positions N and 2N must be flagged. The #pragma linkage instructs the C compiler to create a Fortran-style argument block with the last word flagged. Unfortunately there is no way to set the sign bit also in the middle of the argument block. Attempts to patch the argument list such as

REAL* y_ref = 
    (REAL*)(0x80000000 | (int)&y);
Sub( &x, ch_ptr, y_ref, 
    &x_size, &ch_len, &y_size );

do not succeed. (It seems that C/370 clears the sign bit again before storing the address into the argument block.)

The work-around is to use a wrapper routine which receives the character arguments as Hollerith data and copies them into temporary CHARACTER variables. For example,

#pragma linkage(K77XCX, FORTRAN)
#pragma linkage(SUB, FORTRAN)
extern SUBROUTINE SUB;

SUBROUTINE* sub_ptr = SUB;
K77XCX( &sub_ptr, &x, ch_ptr, &ch_len, &y );

with

SUBROUTINE K77XCX(SUB, X1, K2, L2, X3)
EXTERNAL SUB
PARAMETER(MBIG=256)
CHARACTER*(MBIG) C2
CALL UHTOC(K2, C2, C2)
CALL SUB(X1, C2(:, L2), X3)
CALL FMEMCPY(K2, C2(:, L2))

Obviously, MBIG should be set to the maximum string length ever expected. FMEMCPY is a Fortran callable version of memcp provided by KUIP. The CHARACTER data has to be copied back into the C string in case that it is an output parameter. (UCTOB cannot be used because there the number of bytes copied is rounded up to the next multiple of 4.)

The disadvantage is that each type of argument list requires its own wrapper. This is not foreseen in the cfortran.h approach and is the reason why it cannot be used for C/370.

5- Input/Output

With few exceptions, mixing Fortran and C input/output calls is asking for trouble. Often the runtime libraries of both languages implement separate buffering schemes in order to improve performance.

Terminal Input

One should never mix READ(5,...) and fgets(....,stdin) requests. On most systems it works as long as the input device is really a terminal. However, at the latest when stdin is connected to a file or a pipe the Fortran READ will suck in more than just its share to satisfy the immediate request. Any following fgets will then access the wrong information (or report end-of-file).

Terminal Output

Mixing WRITE(6,...) and fprintf(stdout,...) is safe as long as the output device is the terminal. When stdout is redirected the C stream functions usually start buffering the output.

On a number of Unix systems (RS/6000, Alpha/OSF, Ultrix) the Fortran runtime library does not use the C stream functions, and the Fortran and C output buffers are flushed out of sequence. The C runtime libraries on these systems can be told to limit themselves to line buffering by

    if( !isatty( fileno(stdout) ) )
        setlinebuf( stdout );

It seems that the Fortran runtime libraries are already disciplined enough to use for WRITE(6,...) and PRINT only line buffering. Note that setlinebuf is not a standard Unix function. Hopefully, if it is not contained in stdio.h, it also is not necessary.

Formatted I/O

There is no general way to associate a Fortran logical unit number with a C FILE* stream. KUIP offers C functions ku_read and ku_write as a convenient interface to Fortran I/O statements for reading/writing complete lines. For details refer to the KUIP manual.

(Note that even in Fortran a READ(LUN, '(A)') CHLINE is not portable because the VS-Fortran runtime library is not able to handle V-format files.)

Unformatted I/O

Some systems provide a function getfd to inquire about the C file descriptor associated to a Fortran logical unit number. This is however non-standard, and often Fortran unformatted sequential I/O adds system-dependent control information to allow for variable length records.
The CIO package in KERNLIB provides a Fortran callable set of interface routines to the C read and write functions in order to perform system-independent unformatted I/O, both for sequential and direct access with fixed record sizes.

6. Initialization

On many systems the runtime libraries for a given language has to be initialized — a task which is then performed by the main program.

6.1. Fortran main program

With a Fortran main program calling C functions directly or indirectly the following points have to be respected.

C/370

The C/370 runtime library is automatically initialized when the first C routine is called. The system resources allocated there are released again as soon as the Fortran routine, from which the C routine triggering the initialization was called, returns. Calling another C routine after that does not cause a re-initialization and the program usually crashes because the malloc heap has become invalid.

This effect can be safely avoided by calling the first C routine immediately in the Fortran main program. KERNLIB contains a dummy routine INITC for this purpose.

When executing a module containing C code the C runtime libraries have to be included in the GLOBAL library lists. The CERNLIB command is the most convenient call to this. Note that the C compiler itself is a C program and will abort with an unintelligible message if the libraries are not defined.

VMS

A Fortran program should call VAXC$CRTL_INIT (either from Fortran or from C) before calling any functions from the C runtime library. There can be strange effects if this rule is not obeyed.

6.2. C main program

Our experience with a C main program calling Fortran routines is limited. One known restriction is that the GETARG routine for retrieving the command line arguments in Fortran cannot be used. Calling this routine usually results in unresolved externals called $f77_argc or similar.
7. Program Library News

7.1 Release 94B of the CERN Program Library

CERN Program Library Office CN/ASD

1- Access to CERNLIB material

The preferred method by which CERN Program Library materials are distributed has been via the network for several years now. The vast majority of all requests are now handled this way. Due to manpower constraints, we are no longer able to offer distribution on magnetic tape and all access should now go via the ftp server, asisftp.cern.ch.

To register for access to CERN Program Library materials, connect to asisftp.cern.ch using ftp. Specify anonymous at the Name: prompt and give your full e-mail address as password. Once connected, you will receive a welcome notice, giving the location of the CERNLIB software. You will also find a file cernlib_registration. If you cannot change directory to the cernlib directory, retrieve this form and follow the instructions therein.

If you plan to install the CERNLIB software, either from ready-built libraries and executables or from the source files, your attention is drawn to the installation guide in the cernlib/doc/ps.dir directory (install.ps.gz and install2.ps.gz).

Please note that many of the files in the cernlib tree contain a suffix such as .gz or .Z. These files are compressed using gzip or compress. If you do not have access to these utilities, leave off the suffix when you retrieve the files and they will be automatically uncompressed on the fly.

2- Release 94B of CERNLIB

Release 94B of CERNLIB is currently in preparation. You will see the files appearing in the NEW area over the next few days and weeks. We strongly encourage testing of these files and feedback of any problems.

At the time of writing, we plan to make the release on Tuesday, 18th October.

3- Obsoleted programs

The following programs are declared obsolete with release 94B of the CERN Program Library. The programs will continue to exist until further notice, but new applications should use the suggested replacement and existing applications should be converted wherever possible.

If the eventual deletion of any of the programs or routines mentioned below would be of inconvenience, please contact the CERN Program Library, preferably by e-mail to one of the standard addresses listed on the “if you need help” page in this newsletter.

3.1- ZEBRA RZ conversion utilities

The four RZ conversion utilities RTOA, RTOX, RFRA and RFRX were written at a time when the ZEBRA RZ package did not support exchange mode files. Exchange mode files have been supported by RZ for several years now and are the default for certain applications such as HBOOK, HEPDB and CMZ.

RZ exchange mode files may be transferred between machines using binary ftp, or accessed remotely over NFS or AFS. This is the recommended procedure, and obviates the need for the stand-alone RZ conversion utilities and the corresponding commands in the ZFTP file transfer utility (GETRZ/PUTRZ).

On some systems, notably VMS and VM/CMS, the correct file format must be specified either as an option when transferring the file, or else the file must be converted after the transfer.

VMS systems

On VMS systems, the file must be converted using the SET FILE/ATTRIBUTES=LRL=record_length command. On systems running versions of VMS prior to 6.1, this command is not available and the command file CERN:[PRG.EXE]RESIZE.COM can be used instead.

With version 3.3 (February, 1994) of TGV Multinet TCP/IP software for OpenVMS, it is possible to set the record size for binary transfers in FTP (previously the only allowed sizes were 512 and 2048 bytes). The commands are RECORDSIZE nnmm when the ftp session originates on the openVMS computer, and QUOTE SITE RMS RECSIZE nnmm when ftp is started from the non-VMS end. Here are examples of the two cases.

- For a transfer starting on an OpenVMS computer

  $ FTP SHIFT.CERN.CH
  SHIFT.CERN.CH> USER yourname
  <Password required for yourname.
  Password:
  <User leeviv logged in.
  SHIFT.CERN.CH> VERSION
  DUKPHY.PHY.DUKE.EDU
  MultiNet FTP user process 3.3(109)
  SHIFT.CERN.CH> BINARY
  Type: Image, Structure: File, Mode: Stream
  SHIFT.CERN.CH> RECORD-SIZE 32400
  SHIFT.CERN.CH> RECORD-SIZE
  Record size for IMAGE files: 32400
  SHIFT.CERN.CH> GET myfile.rzhist myfile.rzhist
  SHIFT.CERN.CH> QUIT
  $
For a transfer starting from a non-Multinet computer

    > ftp dukphy.phy.duke.edu
    Connected to dukphy.phy.duke.edu.
    220 DUKPHY.PHY.DUKE.EDU
    MultiNet FTP Server Process 3.3(14)
    at Mon 13-Jun-94
    Name: yourname
    331 User name (yourname) ok.
    Password, please.
    Password:
    ftp> binary
    200 Type I ok.
    ftp> quote site rms recsize 32400
    200 IMAGE file record size now 32400 bytes
    ftp> put myfile.rzhist myfile.rzhist
    ftp> quit

VM/CMS systems

On VM/CMS systems, one can specify the record format of the target file using an ftp subcommand such as the following:

    bin f 3600

If connecting to a VM/CMS system, then use

    bin
    quote site fix 3600

3.2- TELNETG

TELNETG is an enhanced TELNET that permits applications on remote systems to use the graphics features of a local workstation. This package is increasingly difficult to support and frequently requires modification for different systems or compilers. Similar functionality is provided as part of X, which is now widely available and is the recommended method for obtaining such functionality.

3.3- M231

The M231 package provides a number of routines that convert between VAX and IBM floating point formats. These routines exist only in the VMS versions of KERNLIB and are written in MACRO. They support VAX F and D floating point formats only.

The recommended format for data that is to be transferred between systems is that adopted by ZEBRA, namely

- 32 bit data, big endian
- IEEE floating point
- ASCII character set
- Two’s complement

and it is recommended that all future applications use this format for data that is to be transferred or shared between multiple systems.

4- Update on the LUND Monte Carlos

New versions of the various LUND Monte Carlos (JETSET, PYTHIA, ARIADNE, FRITIOF and LUCIAE) will be released with version 94B of the CERNLIB. Each of these packages is made available as a separate library, with the exception of JETSET and PYTHIA, which are distributed in a single libzry.

The exact details are:

4.1- JETSET74

Contains JETSET 7.4 and PYTHIA 5.7

4.2- ARIADNE

Contains ARIADNE 4.05

4.3- FRITIOF

Contains FRITIOF 7.02, JETSET 7.3, PYTHIA 5.5, and ARIADNE 4.02r. The dimensions of some arrays have been increased in PYTHIA and JETSET as compared to those versions available from Torbjorn Sjostrand, and ARIADNE has been optimized for use with FRITIOF.

4.4- LUCIAE

Contains LUCIAE 2.00, JETSET 7.3, PYTHIA 5.5 and ARIADNE 4.02R. As for FRITIOF, small modifications have been made for compatibility and optimization.

5- Apollo DN10K support

We plan to stop building and distributing CERNLIB for Apollo DN10K machines as from the end of this year. (In other words after the forthcoming 94B release.) If this would seriously inconvenience you, please contact the CERN Program Library Office.

6- Fortran compiler versions on various systems

The 94B release will be the last release in which the following compiler versions will be used:

- Sun F77 1.4
- DEC OSF F77 1.3
- AIX XLF 2.3
- VAX/VMS VAX Fortran

7- Apollo, Sun OS and Ultrix support

We tentatively plan to stop building and distributing CERNLIB for the above systems as from the end of 1995. If this would seriously inconvenience you, please contact the CERN Program Library Office.
7.2 A Second Life of the CERN POISSON Program Package

V. I. Klyukhin and B. I. Klochkov, IHEP Moscow, Russia

Introduction

The CERN POISSON package [1] consists of a set of programs designed by R. F. Holsinger and C. Iselin for the solution of Poisson's or Laplace's equation in two-dimensional regions. The package is the most popular to solve magnetostatic problems for the magnetic systems with the geometry reducing to the two-dimensional case. There are four constituents of the package: the mesh generator LATTCR, the equation solver POISCR, the plot program TRIPCR, and the force computing program FORCCR. To discretize the problem a topologically regular triangle mesh is used over all the programs and each node of this mesh is identified by two logical coordinates in the arrays of data and two physical coordinates in the $xy$- or $rz$-plane in which the geometry of magnetic system is described.

The Package Modifications

Our first experience in the use of the POISSON package has come from calculations of the superconducting solenoid magnetic field for a colliding beam setup planned to be constructed at UNK [2]. During this period we modified the program TRIPCR to plot not only magnetic flux

![Figure 7.1: Solenoid magnetic flux density.](image)

...equipotential lines, as was provided by the program, but also the dependence of flux density on one of the physical coordinates for different constant values of another one [3] as shown in Figs. 7.1 and 7.2.

![Figure 7.2: Toroid magnetic flux density.](image)

We became interested in modifying the package while doing the calculations for the ATLAS magnetic system [4, 5]. This resulted in the creation of two interface programs: POISGT [6],

![Figure 7.3: Solenoid magnetic system.](image)

...which extracts a magnetic field map into ZEBRA structure and permits to use it for the GEANT simulations, and LATTCP [7], which is an auto-mesh generator using the physical coordinates only. The latter program enables us to use graphic primitives such as 'LINE', 'CIRC' and 'RECT' to prepare the input data for LATTCR in the most simple way, and to describe the magnetic system configurations such as shown in Figs.
7.3 and 7.4. The program POISSON gives the possibility to calculate the magnetic field integrals [8] which determine the particle momentum resolution in the inner tracker placed in a non-uniform solenoidal field.

Figure 7.5: Tile calorimeter structure.

To solve the problem we had to increase the number of the mesh nodes supported by the package from 40 000 to 270 000, and also to increase the number of regions characterized by the different magnetic properties from 60 to 7000. At CERN we use the modified POISSON program package on the AXCERN, ASF, CSF and PARC computer clusters. A calculation of the magnetic system shown in Figs. 7.3 and 7.5 takes about 3 hours of CPU time on the DEC ALPHA machine. On UNIX machines it is much faster.

Conclusions

The modified POISSON package and the newly designed interface programs enables one to prepare the geometry descriptions of the magnetic systems in a simple way, to increase considerably the number of elements in the magnetic system, to visualize graphically the magnetic field values for different cross-sections of the system, to extract the magnetic field map and to use it in the GEANT simulations, and to calculate magnetic field integrals for the tracking volume placed into uniform solenoidal field.

All these new possibilities make the package well suited to solving a lot of magnetostatic problems for various types of the collidersetup magnetic systems.

Acknowledgements

The authors thank P. Jenni, M. Nessi and A. P. Vorobiev for the support of this work.


7.3 End of my Activities with CERNLIB

René Brun CN/ASD

I would like to announce that I shall be leaving the CN/ASD group and giving up the responsibilities I had until now concerning all CERNLIB related software development.

For many years I have been involved in the development of various software packages and general utilities. I started with the implementation of the HBOOK system in 1974, then the versions 1 and 2 of GEANT as well as many other products. In 1982, I joined the OPAL collaboration and started the implementation of GEANT version 3. In collaboration with several colleagues, I continuously developed the GEANT system over 10 years until version 3.14 in 1991. In 1984, I contributed to the ZEBRA collaboration in implementing the subsystem RZ that became later a main component of several other packages. In 1987 I started the coordination of the PAW project and of the PIAF system in 1993. All these packages have been developed in very close collaboration with a very small team of very talented and experienced programmers and physicists. But the main success of these packages is essentially due to the fantastic feedback and collaboration from a huge number of people in several hundred laboratories or institutions in the world.

I am currently working with my PAW colleagues to polish the 94B release expected in October. This release is a consolidation of many functions and includes also new developments.

We have spent more time than usual in the testing phase and I am sure that 94B will be an excellent release.

I have enjoyed working in this field for many years, not only for the type of work, but also because I have been in contact with so many different people and cultures. I would like to thank all of you who have contributed in many different ways to the success of our work.

7.4 PAW Release (2.05)

The PAW Team CN/ASD

A new version of PAW (2.05) will be released together with CERNLIB 94B. The major new features and enhancements in this version are summarized below. However, as the release is one month after this newsletter, we invite people to refer to the file

/cern/pro/doc/paw.news2050

for complete and up-to-date information concerning bug fixes and improvements since the last release. News will be posted on all systems to announce the exact date of the release and the availability of this file, which will be accessible via anonymous ftp on asistp.

HBOOK

94B is quite a substantial release concerning HBOOK. We refer users to the file paw.news2050 for a more complete description of all corrections, changes and new features.

Bug fixes, protection, or modifications have been made in HDIFF (precision), HINDEX and HINPRX (extend range of formats), HRESETM1 and HERZOM (profile histograms or histograms with error bars), HFCXY (protection for very large real numbers), HALLOC, HBALLOC, HGNT1, HGNT2, HGNTBF (completely re-written), HSCR, HMINUT, HFITHC, HNTNAM, HCDIR, HREND, HFC1, HBOOKN (disk-resident tuples).

New HBOOK routines

SUBROUTINE HBAR2(ID)

This routine can be used to create the data structure to store errors for 2-D histograms (like HBARI for 1-Ds). By default, the errors are set to the sqrt(contents).

The HBOOK routine HPAKE (or the PAW command PUP/ER8) may be used to fill errors. If HBAR2 is not called before HPAKE, then HPAKE invokes HBAR2 automatically. When HBAR2 is called, routines BFILL or BF2 will accumulate the sum of square of weights.

The error bars for 2-Ds are used by the fit routines or the Histo/FIT command. The error bars are not drawn by the HPlot routines.

This routine can be called from the PAW command line or from a COMIS routine.

SUBROUTINE HMERGE(NFILES,FILES,FILOUT)

Merge the NFILES HBOOK files FILES with identical objects and directories into a new file FILOUT.

SUBROUTINE HMERGIN

Same action as HMERGE, but the list of files to be merged and the output file are prompted interactively.

SUBROUTINE HNDUP(ID1,ID2,NEWBUF,CHTITL,CHOPT)

Duplicate definition of an Ntuple. A new Ntuple ID2 is created with the same definitions as ID1, but with 0 entries.

SUBROUTINE HNTYDEF(ID1,IVAR,CHTAG,BLOCK,ILOYEE)

Returns the variable definition as given in HBNAME for variable with index IVAR in Ntuple ID1 (Ntuple must already be in memory).

SUBROUTINE HRENAME(ID,CHL0D,CHNEW)

To rename column CHLD of Ntuple ID into CHNEW.

SUBROUTINE HCONVOL(D1,ID2,ID3,ERROR)

Perform 1-D convolution of ID2 with ID1 as kernel with result in ID3.

HBOOK routines with new options or modified

HLIMAP
New option (Unix only): when the first parameter LIMIT=0, an existing shared memory is attached and becomes the current HBOOK directory.

**HOPERA**
New option 'B' to be given with option 'E' to compute Binomial Errors in the case of division.

**HDOP**
Options PROS or PROE are displayed with option SHOW.

**HHRESET**
Can reset Ntuples as well.

**HHPROF**
New options 'S' and 'I' in order to choose how errors are calculated. **Profile histograms can now be filled with weights.**

**HLINE**
Support for RLOGIN directories (this means all possible combinations of directories are now supported).

**HFITH** and **HFITV**
New option 'K' with option 'M'. When 'KM' is given, the parameters set in Application HMINUIT are not reset.

**HFITH**
HMINUIT has been modified to compute an equivalent chi-square in the case of a log-likelihood fit.

**HFITV**
2-D vectors may now be specified (also apply to PAW command VECTOR/ FIT).

**HMCLNL**
HMCLNL and HMCLNL
For multiple simultaneous fits, HMCLNL has to be called once for each set of histograms and then HMCLNL must be called with the histogram identifiers and number of MC sources, as well as the fractions. Old HMCLNL has been renamed as HMCNCL and will be deleted. HMCNLL already uses the new HMCLNL. HMCLNL and the new and old versions of HMCLNL both contain a banner announcing the change, as it's not backwards compatible.

### New PAW/PAW++ Commands

**NTUPLE/HMERGE OUTFILE INFILES**

**OUTFILE** C 'Output file name' D=''

**INFILES** C 'Input file names' D=' ' Vararg

Ntuples are merged and histograms with identical ID are added.

**NTUPLE/DUPLICATE ID1 ID2 [NEWBUF TITLE OPTION]**

**ID1** C 'Source Ntuple'

**ID2** C 'New Ntuple'

**NEWBUF** I 'Buffer size' D=-1

**TITLE** C 'Title of ID2' D=''

**OPTION** C 'Options' D='A'

Possible OPTION values are:

A Set the Addresses of variables in common /PAWCR4, etc./.

M Create ID2 as a Memory resident Ntuple.

' ' Copy ID1 structure in ID2.

Reset addresses of variables.

The structure of Ntuple ID1 is duplicated in a new Ntuple ID2. This command is useful when one wants to create an Ntuple with the same variables but only a subset of the events. NEWBUF is the buffer size for ID2. If NEWBUF<0 the buffer size of ID1 is taken. If NEWBUF=0 the current buffer size is taken (10000 words for RWN). NEWBUF>0 will be the new buffer size. If TITLE=' ', ID2 has the same title as ID1. In the case of a disk-resident Ntuple (default), ID2 is created in the current working directory which must be open in WRITE mode.

**KUIP/BUGREPORT**
Send bug report to the PAW development team by e-mail. The local editor is automatically invoked with a template bug report. After the template has been edited, full version information about the paw version and the operating system is appended. The user is asked for a confirmation before the report is send. (N.B. The command is not yet implemented on all systems.)

In Paw++ this command can be accessed from the 'Help' menu of the 'Executive Window' (menu item 'Mail Paw++ Developers').

### New Options in Existing PAW Commands

**NTUPLE/UWFUNC**

**NTUPLE/UWFUNC IDN FNAM [ CHOPT ]**

**IDN** C 'Ntuple Identifier'

**FNAM** C 'File name'

**CHOPT** C 'Options' D=''

Possible CHOPT values are:

' ' Generate the FORTRAN skeleton of a selection function.

E Present the selection function in the local editor.

To generate the FORTRAN skeleton of a selection function or the INCLUDE file with the columns declaration.

This command is now able to generate include files. A FORTRAN function is generated if the FNAM is of the form, xxx.f, xxx.for, xxx.fortran. Otherwise an INCLUDE file is generated.

### NTUPLE/SCAN

**NTUPLE/SCAN IDN [UWFUNC NEVT IFRST OPT VARLIS]**

**IDN** C 'Ntuple Identifier'

**UWFUNC** C 'User cut function' D='0'

**NEVT** I 'Number of events' D=99999999

**IFIRST** I 'First event' D=1

**OPT** C 'Options' D=''

**VARLIS** C 'Names of the NVARS variables to scan' D=' ' Vararg
Possible OPT (options) values are:

S  Graphical scan (spider plot).
'  Alphanumeric output of the Ntuple.

New option 'S' for a "graphical scan" (spider plot) and new way to specify the variables list (parameter VARLIS) (see paw.news2050 for more details.)

HISTOGRAM/FIT
New option 'K' in order to keep the settings of "Application HMINUIT" for a subsequent command. New option 'M' in PAW++ in order to access a specific "MINUIT fit panel". This new tool allows one to define interactively and graphically the initial value of the fit parameters and to start, afterward, the minimization process.

HISTOGRAM/CREATE/PROFILE
New options 'S' and 'I' in order to choose how errors are calculated.

HISTOGRAM/GET_VEC/REBIN
New option 'N' (do not normalize values in Y)

HISTOGRAM/ PLOT:
- New options 'FB' and 'BB' to remove the Front and Back Boxes on surface and lego plots.
  Example:
  PAW > H/PLOT 20 SURF,FB
  plots the histogram 20 as a surface without the front box.
- Option 'A' (to avoid the axis drawing) works also on 2-D histograms representations.

LECT, VLECT and CUTS
these three commands have the workstation identifier as an additional and optional parameter in order to allow request locator in any window.

PIAF (Ntuples)
The command NETWORK/PIAF/STATUS, to inquire about the status of the Piaf server, has been modified to print the average load statistics. The termination of a Piaf session has been improved. Problems have been corrected in file opening and in the case of interruption.

COMIS
Apart from various bug fixes, the main enhancements in COMIS are:

- Upper and lower cases are allowed in format fields,
- It is now possible to call KUVECT for vector's name in PIAF slave processes,
- Dynamic loading is possible on AIX, SUN and SGI (it was already available on HP).

To get a full list of all routines (from CERNLIB) which are now callable from COMIS type HELP CALL.

New Histogram Title Scheme (PAW/HPLOT)

Histogram titles may now have the following form:

'Histogram_title ; XXX ; YYY ; ZZZ'

where 'XXX', 'YYY' and 'ZZZ' are respectively the X, Y and Z axis titles. In this model, one field can be blank. For example, an histogram title like

' ; XXX ; YYY'

will remove the histogram title but will leave the titles along the X and Y axis.

The command ATITLE has been updated according to this new scheme (additional optional parameter ATM):

* GRAPHICS/HPLOT/ATITLE [ XTIT YTIT ZTIT ]

XTIT C 'X Axis title' D=''
YTIT C 'Y Axis title' D=''
ZTIT C 'Z Axis title' D=''

Draw axis titles on the axes of the present plot zone.

HIGZ/HPLOT

Histogram drawing:
- Improvements have been made to allow the drawing of histograms with a large number of bins.
- Axis tick marks are redrew if they have been erased with an histogram drawn with option ‘S’ as a filled area.

Surface drawing:

When all the bins (not a subrange) of a histogram are drawn as a surface (with the options POL, CYL, SPH or PS) the last X bin is connect to the first one. In the previous versions an unfortunate small gap was visible.

Optimization in axis drawing:
- to avoid identical labels on the same axis in the case of very small ranges,
- to avoid x10^-0.

PostScript driver optimization:
- Polymarkers and line drawing have been optimized in order to reduce the size of the PostScript files.
- The cipping is now done via the "clip" PostScript directive. This solves some fill area clipping problems, and it improves the speed.
- Better size mapping between X11 and PS fonts.
New KUIP Command

- **KUIP/SET_SHOW/DOLLAR [ OPTION ]**
  
  **OPTION** C 'Substitution ... D=?’
  (... 'Substitution of environment variables')

Possible OPTION values are:

?       show current setting
ON      enable substitution
OFF     disable substitution

This command allows one to enable/disable the interpretation of environment variables in command lines. The startup value is ‘ON’, i.e. ‘‘$var’’ is substituted by the variable value.

Note that the system function ‘‘$ENV(var)’’ allows the use of environment variables even for ‘DOLLAR OFF’.

New KUIP System Functions

- **EXEC** ('macro args')
  returns the EXITM value of EXEC call,
- **FEXIST** (filename)
  returns 1 if file exists or 0 otherwise,
- **SHELL** (for Unix only):
  - **SHELL(cmd,N)**: returns the N’th line of shell command output
  - **SHELL(cmd,sep)**: shell output with newlines replaced ‘sep’
  - **SHELL(cmd)**: same as SHELL(cmd,’ ’)
- **ENV** (var)
  to get environment variable independent of DOLLAR setting (cf. new command KUIP/SET_SHOW/DOLLAR).
- **CALL**, **ICALL**, and **DCALL**
  allow one to call REAL, INTEGER, and DOUBLE PRECISION functions, respectively. The function call must be enclosed in quotes, for example:
  - **CALL(‘fun.f(1.5)’)**

To get a full list of all KUIP systems functions you can type HELP_FUNCTIONS.

KUIP/PAW Miscellaneous

- On Unix systems, depending on the value given to ‘‘HOST_SHELL’’ all the aliases and variables defined in the login file (e.g. .kshrc for the “ksh” shell) can be used.
- ‘‘MACRO/DEFAULT -AutoReverse’’ can be used without running LAST.KUMAC at startup-time.
- For PAWPP on VAX/VMS the default editor has been set to ‘‘edit/tpu/disp=decw’’.

KUIP new macro interpreter

The macro interpreter (KUMAC) has been re-written in C. The first difference is that ‘‘EXEC’’ inside a macro is treated like any other command. This solves a number of outstanding problems:

- ‘‘EXEC’, ‘‘M/EXEC’’, or any other abbreviation of the command path ‘‘MACRO/EXEC’’ is now allowed.
- Before this would lead to undefined behavior due to a recursive Fortran call.
- Defining an alias for an EXEC call or changing the MACRO/DEFAULT path during macro execution has now the expected effect. Before aliases and .kumac searches were done at macro compilation time rather than execution time.
- It is now possible to execute a macro containing EXEC calls to non-existing .kumac files — provided, of course, that the EXEC is never reached, e.g. in a ‘‘CASE $OS IN ...’’ construct.
- It is now possible to create or to modify a .kumac file during macro execution and call the macros in the new file.
- Mixed-case path names of .kumac files are possible now.

The response time should be better than in previous versions:

- The new macro compiler needs only one pass rather than two passes over the .kumac file.
- A macro calling other macros will compile the other .kumac files when needed rather than compiling everything up-front.
- Macros are cached — if a .kumac file has been invoked already beforehand and it hasn’t been changed then it is not recompiled when it is used again.

In addition the following bugs are fixed in the new version:

- **ON ERROR GOTO** works now also for nested EXEC calls.
- **Undefined numbered variables** are now set to ‘’ as documented.
- Depending on the length of the .kumac file name the use of several variables in an expression could lead to truncation due to the fixed length of a Fortran character string.

The new version provides the following new features:

- The **READ statement** allows one to specify the prompt string. It shows the default value also if prompt is user supplied.
- The **NEXTL statement** allows one to continue with the next loop iteration, similar to the ‘‘continue’’ statement in C.
- **BREAKL and NEXTL** allow one to specify how many levels of nested loops should be skipped.
- The **STOPM statement** allows one to stop macro execution, i.e. unwind nested EXEC calls and return to the command line prompt immediately.
- The **RETURN** statement allows one to specify the return value.
- The variable ‘‘[0]’’ contains the fully qualified macro name, i.e. ‘‘/path/file.kumac@macro’’.
- **ON ERROR** allows now the following choices:
  - **ON ERROR CONTINUE**
  - **ON ERROR GOTO label**
  - **ON ERROR EXITM value**
  - **ON ERROR STOPM**
The spelling "OFF ERROR" instead of "OF ERROR" is now allowed.

* Block constructs can now be written on a single line using ";" as line separator, e.g. "cmd1 ; IF ... THEN ; cmd2 ; ENDIF;"

* The macro constructs are now documented in the online help — try "HELP SYNTAX". Thanks to Mike Kelsey for editing the text.

* "!!" as macro argument will now use the default value given in the MACRO definition.

but it also contains the following known incompatibilities:

* The command /MACRO/RECURSION has been deleted — all macros are recursive now.

* "(:=") as alternative to the assignment operator "=" is not supported anymore. Note that "LET var = expr" is still allowed.

Special thanks are addressed to Robert Franchisseur, Mike Kelsey, and Andrea Parri for beta-testing the new macro inter-preter.

**KUIP/Motif ("Panel" interface)**

The "panel" interface built inside KUIP/Motif for user-defined panels has been completely redesigned. It should be fully compatible with its previous implementation but a lot more powerful: it is now possible to define the panel keys (or buttons) with labels (aliases), exact KUIP commands or icons (pixmaps). The general KUIP command for a panel key definition in KUIP/Motif is now:

```
panel x.y command [label] [pixmap] [option]
```

x.y is the key position (column and row number).

command is the complete command (or list of commands) to be executed when the corresponding button is pressed.

label (optional) If specified, this text is used for the button label (when the appropriate "View" option is selected) instead of the complete command (which is generally too long for a "user-friendly" button label).

pixmap (optional) to be specified for graphical keys. If a <pixmap> name is given the graphical representation is displayed by default.

option (optional) a set of options to provide different behavior for button handling (e.g. "T" for toggle behavior). This is not yet finalized.

All parameters inside "[]" are optional. But as their position is meaningful one should replace any missing parameter by a "[]" if it is followed by a given one.

It is possible at run time to change the "panel visualization" by selecting the appropriate entry in the menu-bar entry "View" ("By Name", "By Icon", "By Name and Icon" (not yet implemented), "By Command (normal)" and "By Command (1 col.").

The application can define its own icons (pixmaps) either in the command definition file (CDF) with the directive "Icon_bitmaps" or the user himself can do it at run-time (or in a KUIP macro) with the command:

```
/MOTIF/ICON pixmap filename
```

pixmap is the name given to the icon bitmap and used in the "panel" command (described above).

filename is the name of the file where the icon bitmap data are stored.

To create a new icon bitmap (or pixmap) one can use the X11 standard bitmap editor "bitmap". E.g., to get a 20x20 pixel icon called m1 one can execute

```
bmp m1.bmp 20x20
```

The output file m1.bmp starting with "define m1_width 20 ..." has to be referenced in the command "/MOTIF/ICON m1/user/.../m1.bmp" or can be put in the command definition file (CDF).

It is also possible to define a "palette of panels" where several KUIP/Motif user-defined panels of commands can be grouped together into one new KUIP/Motif container widget (keeping the same panel definitions) with the new KUIP/Motif command (available in PAW++ and all KUIP based application):

```
MULTI_PANEL [title] [geometry]
```

E.g. the command

```
MULTI_PANEL 'My Palette' '200x100+0+0'
```

will display a palette, or "multi_panel" widget, with the given title and geometry. After this command execution all panel definitions and executions will go into that widget, in the form of a label (containing the panel title) and an "arrow button", which can be pressed to display the panel contents. To end a "multi-panel" setting one just has to issue the command:

```
MULTI_PANEL end
```

N.B. For users who are familiar with the "UIMX" User Interface Management System, the KUIP/Motif "multi_panel" is an emulation of the "Palette" widget which is built into this program.

**KUIP/Motif C callable interface**

A C callable interface for KUIP/Motif application programmers to perform various actions (e.g. panel and palette setting at initialization time) is accessible. It provides high-level functionalities without the application programmer having to be involved in direct Motif/Xlib programming. This can be extended according to user requests. Full documentation is available in the latest edition of the KUIP manual.

**A word of Thanks**

The PAW team would like to thank all those who have contributed to the testing phase of this release of the PAW system.