DESY Computing Newsletter
No. 4 February 1993

A Publication of the DESY User Support Group

Contents

Editorial

by Michael Behrens

This newsletter mainly covers topics related to Unix and distributed computing at DESY. These topics - even if not too closely related in principle - turn out in practice to be closely tied together. Many questions on how to register on the central Unix services and how to use them for regular work are asked at the User Consulting Office, and there is a real need for documentation and instruction.

While one can access a Unix host from almost any type of terminal, X–terminals are becoming the most common way of accessing Unix (and other systems) at DESY. One article in the Newsletter covers the basics of X-terminal usage at DESY. The articles on editing, using TEX, and backup on Unix should help you with your first steps in this new world of Unix.

The article on computer accessible information goes beyond Unix and tries to cover all platforms available at DESY. The article on the QMS printers is mainly of interest to users of the DESY IBM, and finally the article on accounting is for IBM users only.

For this issue we have found a fair number of people besides the editors, who contributed articles for the newsletter. We greatly appreciate this development and we want to thank all authors for their work. We hope that this trend will continue in the future.

A few Words on Printing

Printing is an area that has already been undergoing significant changes for some time. Instead of a small number of large printers locally attached to a certain host, DESY has moved to a large number of smaller printers with graphic capabilities distributed all over the DESY site and accessible over the network from (almost) any host. The number of network printers currently exceeds 100, and managing this large number in a coherent way is a demanding task.

The Computer Centre is working on streamlining the printing service and on providing suitable documentation. This documentation is currently being prepared, but it would exceed the scope of this newsletter if we included it here. The full description of printing services will therefore become a document of its own when it is completed.

Many people at DESY are concerned about the amount of natural resources consumed by the large volume of printing at DESY. To minimize the negative impact, both the users and the computer centre staff have to work together. Double sided printing will be available wherever possible to reduce the consumption of paper. We also try too to use paper that is produced with the smallest possible impact on nature while still allowing reliable operation of the various printers.

In the end it is up to you to restrict the number of pages printed and to accept less glossy paper for every day usage. It is good practice to only use high cost and high quality printout when they are really needed. Colour plots should only be used after a successful test with a normal black-and-white paper copy.

Editors of the DCN are:		
Katherine Wipf	R01WIP	ext. 3222
Michael Behrens	R01BEH	ext. 2556
Jan Hendrik Peters	R01NET	ext. 2583

• DESY Unix Activities

by Wolfgang Friebel

At DESY there are a wide variety of computers running under a UNIX operating system. The main flavours of UNIX found at DESY are the operating systems HP–UX (HP), Domain OS (Apollo), AIX (IBM), ULTRIX (DEC), IRIX (Silicon Graphics), Sun OS (Sun) and Convex OS (Convex).
To coordinate the various activities around these operating systems, a DESY UNIX committee (DUX) was formed in August 1992. Presently it consists of about 15 people both from the system management groups and from the physics groups at DESY. The chairman is W.Friebel, Zeuthen.
One of the main tasks of the committee is to provide a uniform user environment for UNIX, independent of the underlying hardware. Various topics have been discussed in the meetings to achieve this goal. As of December 92 the following proposals were made by the DUX:

- Support for a csh-like login shell (tcsh) and a bourne/korn shell-like login shell (zsh) by the computing centre group
- Usage of a set of environment variables common to all systems
- Creation of login procedures which run on all supported platforms
- Uniform X11 environment based on tools developed by T.Finnern
- Keyboard mappings for some frequently used tools (emacs, less, 3270-emulation)

Much more is planned in this field, including coordination with HEPIX activities on the same subject.

The most important fields to be discussed in the near future are a centralized user registration, Network Information Services (NIS, Yellow pages) for DESY wide global logins, and support for system installation and software distribution.

The minutes of the DUX meetings as well as other documents are accessible by anonymous ftp from

`ftp.ifh.de`

in the directory

`pub/dux`

The members of the DUX can be addressed by the mailing list dux@ifh.de or individually by mailing to `friebel@ifh.de`.
People interested in DUX activities should look at the above mentioned minutes and also follow the discussions in the Netnews group hepnet.hepix.

• How to use X-Terminals

by Jan Hendrik Peters

With the delivery of a large number of X-terminals late last year, everybody now has access to these new terminals. But at this point the problems already start. How can you login? How can you use your old services? What kind of new services are opened up by this new technology?

Step 1: Get an Account

The first problem you encounter is that you need a new account in order to make use of the X-terminal (unlike with the Falco terminals which you just needed to plug in and go). The X-terminals are currently directly connected to either the Apollo-Cluster, HP-Cluster, or the new SGI-Cluster called x4u (X-for-you), which in future will be the hub for most new X-terminals. If you are one of the lucky people who has an X-terminal in his/her office, you just need an account on the cluster this X-terminal is attached to. The name of the cluster a specific X-terminal is attached to can be read off the login screen of the X-terminal. Everybody else should at least get an account on the HP-Cluster and the APOLLO-Cluster. In future you might have to get an account on the x4u-Cluster as well (unless we get a common registry for all machines earlier). Account registration forms can be obtained either from your group administrator or from the UCO. You have to fill out your name, group, location, your requested login id (userid, account name), an initial password and you have to select a default shell you want to work with (more on shells below). With your signature you agree to use your account only for work related to DESY activities and to follow some basic rules on password security, software copyrights, etc. Your group administrator certifies with his signature that you are a member of his group and have the right to work on the various platforms you have chosen. The completed form should be returned to the UCO.

The system administrators of the UNIX systems are:

Ernst-Ludwig Bohnen	HP-Cluster
Thomas Finnern	Apollo-Cluster
Bernd Hellwig	x4u-Cluster

When they have registered you on their cluster, you are ready to go.

A few words about the selection of a login id: With the rather complex situation regarding operating systems and machine types on the DESY site, there is some need for a convention for login ids. On the IBM we have the well established scheme of choosing a login id consisting of 6 characters with 3 characters for the group and 3 for the person (ggguuu type of userid, e.g. f14jhp). On the VAX people everwhere in the world register under their name, e.g. their last name, first name, or combinations thereof up to a length of 12 characters. On some UNIX machines the maximum length for login ids is 8 characters. From the point of view of networking services (ftp, rlogin, rsh, rcp, etc.) it is neccessary to have the same login id on as many systems as possible to fully utilize the power of these commands. Therefore we encourage every user to select his/her login id according to the following scheme: Your login id on the VAX and on the UNIX systems at DESY should be identical with a maximum length of 8 characters. You should select your last name with initials from your first name (e.g. petersjh, mbehrens), or if your name is unique on the site just your last name. The usage of your IBM login id on all platforms is permitted but not encouraged.

Step 2: How to log in

If you have never worked with a mouse or a window management system, the following terminology might prove useful for you:

Motif Button A small display box within the borders of a window.

Mouse Pointer An arrow or cross on the screen indicating the current position for selecting or clicking.

Mouse Button The three buttons on the top of the mouse. The left button is normally used for clicking or selecting, the middle button for positioning text you have selected (cut and paste operation), and the function of the right button always depends on the application.

Clicking Quickly pressing and releasing the mouse button without moving the mouse pointer.

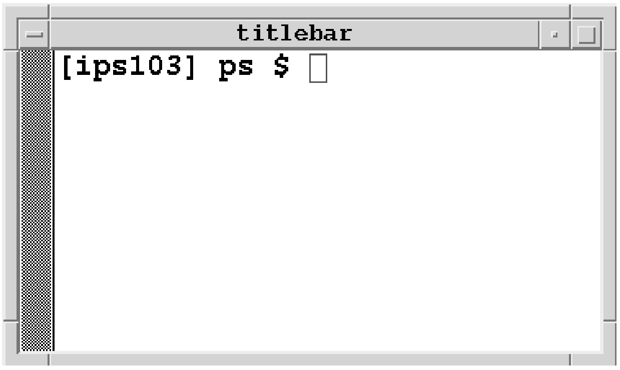
Dragging Pressing and holding the mouse button while moving the mouse pointer on the screen.

If you sit in front of the screen you will see a login window created with the Motif Window Manager (mwm). Motif windows normally consist of input and output fields and buttons. On the login panel on the APOLLO and HP clusters you will find two input fields for your login id and your password and four buttons labeled *OK, Clear, Options, Help.* The buttons are activated by clicking with the left mouse button while pointing to the Motif button. To log in you just type your login id in the upper field, press *Return*, type your password in the second field, and press *Return*. If you receive an "invalid password" message and you are sure that you used the correct password, first click the *OK* button on the message window then the *Clear* button on the login window, make sure the CapsLock key was not pressed and restart the procedure. This should help in almost all cases. The Motif input fields are very sensitive to extra blanks and control characters.

The login panel should now disappear and after a while (depending on the cluster load) new windows will appear. In the right hand corner you will see the *Session* panel with information about your login id, login machine, X-server machine, and the name or IP-address of your X-terminal. The two buttons below allow you to totally logout from the cluster or to temporarily lock the screen. The latter option should only be used if you are leaving the X-terminal for a very short time. If you stay away for a longer time you should logout to allow other people to use the X-terminal. The maximum locking time is currently restricted to 20 minutes and might be reduced in future.

On the left hand bottom of the screen little *icons* will show up which can be activated by double clicking them with the left mouse button or by a single click on the icon and on the *Restore* button in the pop-up menu. (Remark: double clicking only works, if the the clicks occur within a very short time interval!)

In the middle of the screen the login window on your server machine will appear (see picture below). Here you can enter any UNIX command. Depending on the machine the prompt may vary, usually displaying the machine you are on and the directory you are in. This window (as any other *xterm* window) has various parts that can be activated with the left mouse button. With the mouse pointer on the frame, the size of the window can be modified by dragging one of the frame parts. By holding the left mouse button with the mouse pointer on the title bar, the window can be moved around on the screen. By clicking on the minus sign button on the left upper corner a little menu will be displayed, the dot button in the right upper corner will iconize the window and the button with the square on it will maximize the window to screen size.



If you have more than one window, only one of them is active for input from the keyboard (the frame of the active window ususally has a different colour). To activate another window just click on it with the left mouse button.

The big grey area behind all windows is called *root window*. If you press any mouse button on the root window, different root menus will show up. By dragging the mouse to one of the fields the corresponding action can be triggered. The most important aspects cover the creation of new windows on the cluster, on the IBM or VAX.

Step 3: Getting some Work done

While at the beginning you might be tempted to only use your X-terminal for logging into your well known old machines, the UNIX platforms offer you an astounding variety of very nice tools (like grep, awk, pipes just to name a few which I find very useful – not to mention the easy way to produce LaTeXdocumentation and the powerful previewer xdvi (for TeX) and ghostview (for PostScript)).

If you are not familiar with UNIX, you first need to get to know some of the basic commands, get used to the shell, and learn a new editor. The DUX committee (see article by Wolfgang Friebel) encourages you to learn the editor called *emacs*. This is the most widely used editor in the HEP community on UNIX platforms and you can also use it under VMS. A common *emacs* environment for all UNIX machines at DESY (both in Hamburg and in Zeuthen) is under preparation. In this newsletter you will find a short introduction to emacs. The UCO offers a short reference card for emacs and for the basic UNIX editor *vi*. A simple to use editor called *pico* is available on most centrally supported machines at DESY.

A word on shells: The shell is your working environment. It interprets your commands and runs your shell scripts (i.e. clists, procedures, etc). There are two basic families: the C-like shells (csh, tcsh) and Bourne-like shells (sh, ksh, zsh). As with all user environments it is a matter of taste which family of shells you prefer. The DUX suggests that you use a high level shell for your interactive work (e.g. zsh, tcsh) and write shell scripts in a lower level shell (sh). On your user registration form you are asked to name a shell you want to work with interactively. If you don't know any of them, just mark default and you will be supplied with the current default shell (One of Bourne shell family for central UNIX computers). The current top of the list shells are zsh and tcsh (see article by Karsten Künne). Both of them support command history recall (using the cursor keys), inline command editing (cursor and delete keys), and command and file name completion (using the tab key), which are more or less essential for a decent work environment.

What are the most basic commands you will need at the beginning? (Warning: UNIX commands and file names are case sensitive!) :

pwd	print (=show) current working directory
cd *name*	change to directory *name*
mkdir *name*	make directory *name*
vi *file*	edit *file* with vi-editor
emacs *file*	edit *file* with emacs
cp	copy files
mv	rename (=move) files
rm	delete (=remove) files
ls -al	list all files in directory
cat	list contents of file
more	list file page by page
less	list file page by page
lp	print file on printer
chmod	change file access rights
grep	find strings (regular expressions)
echo $NAME	show value of variable NAME
export NAME=value	set variable in Bourne shells
setenv name value	set variable in c shells
f77	invoke the fortran compiler
cc	invoke the c compiler
who	list all user logged in
man	get help for UNIX commands
news	display system news and DESY specific helps

There are a few more concepts which help you get along with UNIX. One of them is redirection. Any command or program needs input and output files. The standard files are stdin and stdout (keyboard and terminal if you are working interactively). To use other files you have to redirect them in the following manner:

cmd <*infile* >*outfile*

or

cmd <*infile* >>*outfile*

where the arrows denote the in (<) or out (>) direction and the double arrow appends to the end of the file.

All UNIX commands normally use stdin and stdout. The output of one command can be used as an input for the next command. This is called piping, the symbol for it is the vertical bar |, e.g.:
who | grep petersjh
(find user entry for petersjh)

Another concept is the foreground and background execution of commands. If a command is executed in foreground, further input from your keyboard might be blocked. To execute a command or program in foreground just give its name, e.g. :
prompt> *cmd*
To execute in background you have to type an ampersand (&) behind the command name:
prompt> *cmd* &
The system will then give you a response with the current job number and the id of the process (pid) you just created:
`[1] 27122`

Here are some commands to control processes:

jobs	list all your background processes
ps	list all your processes
kill %n	cancel (=kill) job number n
fg %n	move job n to foreground
nice	reduce priority of command execution

Step 4: Connections to other systems

Quite often you would like to connect to other computer systems and run X-applications from there. To do so the current X environment variables have to be exported to the remote node. The most important ones in this game are TERM and DISPLAY and the MIT-Magic-Cookie authorization to allow remote access to your X-terminal. If these are incorrectly set, you might not be able to call an editor or to display anything on your X-terminal. R2 has developed some commands to properly transport these variables. Some of these are xrsh, xrlogin, etc (see *news x11 | less*). These commands are or will be installed on all UNIX machines at DESY Hamburg and at Zeuthen to allow easy work on all UNIX platforms.

There is a big class of commands that allow access and command execution on other systems. These *remote* service commands all start with the letter *r* like: rlogin (login to remote system), remsh (APOLLO and HP – execute command on remote node), rsh (SGI – execute command on remote node), rcp (copy file from/to remote system).

However, if you are working with an X-terminal, you should not use rlogin and the remote-shell commands but rather the corresponding X-command.

• UNIX shells

by Karsten Künne

The first environment you interact with when you log in to a UNIX system is the shell. The shell is the interface between you and the system which accepts your input and executes the appropriate commands. One of the big advantages of UNIX is that you can choose between several different shells, you could even write your very own private shell and use it.

In order to choose an appropriate working environment for myself, I compared various widely available shells.
The candidates were:

sh	(Bourne Shell) The Grand-daddy of all shells, the first UNIX shell, written by Steve Bourne.
csh	The shell from the BSD UNIX Distribution, written at the University of California, Berkeley.
ksh	(Korn Shell) A successor of the Bourne Shell from AT&T, written by David Korn.
tcsh	A very popular "free" shell.
bash	(Bourne Again Shell) The shell from the GNU Project UNIX, also a free shell.
zsh	Another very good free shell, written by Paul Falstad.
rc	The shell for the Plan 9 Project, a possible successor of UNIX, written by Tom Duff at AT&T.

From the syntax point of view most shells fall into one of two main groups: shells with sh-like syntax and shells with csh-like syntax. sh-like syntax and csh-like syntax are not compatible. The first group contains sh, ksh, bash and zsh whereas the second group contains csh and tcsh. The rc shell has a different syntax which does not fit into either of the groups.

What a shell should do

What can one expect from a shell? What are the main requirements an interactive shell has to meet?:

1. **Execution of commands:** The shell should execute commands.
2. **Pipes:** The ability to concatenate commands such that the output of one command is used as input for the next.
3. **Redirection:** This means redirecting command input/output from/to files instead of standard input (keyboard) and output (screen).
4. **Configurable Environment:** The ability to define a personal working environment using variables.
5. **History:** A history of executed commands and the possibility to re-execute commands.
6. **Globbing:** The expansion of wild-cards in filenames.
7. **Aliases:** The ability to use aliases for commands.
8. **Functions:** The ability to use shell functions.
9. **Command Line Editing** The ability to edit a command line before execution using the arrow keys.
10. **Job Control:** Report of status changes for background jobs and track-keeping of background jobs.
11. **Completion:** The completion of a partially typed command or file name.
12. **Spelling Correction:** The correction of spelling errors in commands and file names.

Additional demands exist especially for the non-interactive use of a shell (i.e. for shell scripts):

13. **Execution Control:** Loops and conditional expressions are required.
14. **Signal Handling:** The ability to establish handlers to catch signals and perform specific actions.
15. **Arithmetic:** The ability to do arithmetic computations.

Free Shells

The vendor supplied shells sh, csh and ksh should be well known to most UNIX users, but how do the free shells compare to these standard shells?

tcsh

The tcsh is basically a csh with a lot of enhancements. It behaves exactly like the csh, except for the added utilities. The most important of these utilities are :

- Command line editing using Emacs-style commands.
- Visual step up/down and searching through the command history list.
- Programmable command, file name, variable name, and user name completion.
- One can give a command to produce a file/directory/user list in the middle of a typed command.
- Spelling correction of command, file, and user names.
- Enhanced history mechanism.
- An addition to the syntax of filenames to access entries in the directory stack.
- In file expressions one can specify negation of a set of characters or a globbing pattern.

There are more enhancements built into tcsh but these are the most important ones.

bash

The bash is basically a sh with a lot of enhancements and features from (t)csh and ksh and some bash specific features. The bash specific features are :

- Command line editing with arrow keys, configurable.
- Commands `set` and `help` available.
- Additional startup files and shell variables.

Apart from this bash is comparable to ksh.

zsh

The zsh is a very powerful shell. The grammar of zsh is very close to ksh/sh, with csh additions. Zsh contains most features of ksh, bash, and tcsh. Some of the important zsh features are :

- A shorthand for loops. Example:
  `for i (*.c) echo $i`
- A directory stack exists that is accessible with the `dirs` command and `=number`.
- Process substitution. Example:
  `vi =(`*cmd*`)`
  starts `vi` on the output of *cmd*
- Generalized pipes. Example:

  `ls foo >>(`*cmd1*`) 2>>(`*cmd2*`)`

  pipes `stdout` to cmd1 and `stderr` to cmd2.
- Advanced globbing. Examples:

  `ls **/file`

  searches recursively for “file” in subdirectories.

  `ls file<20->`

  matches file20, file30, file100 etc.

  `ls *.(c|pro)`

  matches *.c and *.pro.

  `ls *(R)`

  matches only world readable files.

  `ls *.c~lex.c`

  matches all .c files except lex.c.
- Null command short-hands. Examples:

`< file`	is the same as	`more < file`
`> file`	is the same as	`cat > file`
`>> file`	is the same as	`cat >> file`

- Automatic file stream teeing (redirect to two different output files). Example:

  `ls >foo >bar`

  puts output in two places.

- Incremental history search.

- With the `autocd` option, typing a directory name by itself is the same as typing `cd dirname`.

- Menu completion. Pressing TAB repeatedly cycles through the possible matches.

- Incremental path hashing.

- With `histverify` option, performing csh-style history expansions causes the input line to be brought up for editing instead of being executed.

- Auto-loaded functions (loaded from a file when they are first referenced).

- Generalized argument completion including command name completion, filename and path completion, hostname completion, key binding completion, option completion, variable name completion, and user-specified keyword completion.

- Various nested startup files.

- "which –a cmd" lists all occurrences of "cmd" in path.

rc

The rc shell is a bit exotic compared to the other shells. The syntax is similar to sh, but more based on awk and C. The most important new features compared to sh are:

- Semantic simplifications.

- Parser is based on yacc. This leads to an exact grammar.

- One pass scanning of input stream.

- Signal handling through functions with the signal name.

- Advanced redirection. Examples:

  `cmd >[2] ...`

  redirect file descriptor 2.

  `cmd >[2=1] ...`

  replace file descriptor 2 by a copy of descriptor 1

  `cmd |[2] ...`

  pipe file descriptor 2 into another command.

- Simpler command substitution Example:

  `echo '`{*cmd*}

  substitutes output of *cmd* as parameter for echo.

- Array variables.

- Concatenation operator. Example:

  `echo (a b c) ^ (1 2 3)` is the same as
  `echo a1 b2 c3`

Comparison

For a short comparison of the different shells the following matrix gives a good overview. The numbers on the left side are the numbers of the requirements at the beginning of this article.

	sh	csh	ksh	tcsh	bash	zsh	rc
1	+	+	+	+	+	+	+
2	+	+	+	+	+	++	++
3	+	−	+	−	+	++	++
4	+	+	+	+	+	+	+
5	−−	+	+	+	+	+	+
6	+	+	+	+	+	++	+
7	−−	+	+	+	+	+	−−
8	+	−−	+	−−	+	+	+
9	−−	−−	+	++	++	++	++
10	−−	+	+	+	+	+	−−
11	−−	−−	+	++	+	++	−−
12	−−	−−	−−	+	−−	+	−−
13	+	+	+	+	+	+	+
14	+	−	+	−	+	+	+
15	−−	+	+	+	+	+	+

++ means "very good support"

+ means "good support"

– means "limited support"

– – means "no support"

Conclusion

As one can see from the comparison above, some excellent "free" shells exist besides the vendor supplied ones which compete very well. In general these "free" shells have a lot of enhancements which can really improve the daily work with a UNIX system and could increase the general acceptance level of UNIX.

For me the absolute winner in the comparison is zsh. It's really the most powerful shell I have ever used. I had tried tcsh and bash before, but now I would recommend zsh. I have now been using zsh for several months and have no problems at all. Some of the zsh features are really nice and made my life with UNIX easier.

• TEX on the Apollo-, HP-, and SGI-clusters

by Peter K. Schilling

Release Information of various TEX products as of April 14, 1993			
Program	Apollo-Cluster	HP-Cluster	SGI (x4u)
`tex`	`format=plain 92.2.12` bi-lingual	`format=plain 92.2.12` bi-lingual	`format=plain 92.10.27` tri-lingual
`amstex`	`format=amstex 92.2.12` Version 1.1c bi-lingual	`format=amstex 92.2.12` Version 1.1c bi-lingual	`format=aplain 92.10.27` Version 2.0 tri-lingual
`latex`	`format=lplain 92.2.12` LATEX 2.09 ⟨13 Jun 89⟩ bi-lingual	`format=lplain 92.2.24` LATEX 2.09 ⟨14 Jan 92⟩ bi-lingual	`format=lplain 92.10.27` LATEX 2.09 ⟨14 Jan 92⟩ NFSS, tri-lingual
`latex9`	`format=lplain 92.10.30` LATEX 2.09 ⟨14 Jan 92⟩ NFSS, tri-lingual, (`news latex9`)	———	———
`slitex`	`format=splain 92.2.12` SLITEX 2.09 ⟨10 Nov 86⟩ bi-lingual	`format=splain 92.2.24` SLITEX 2.09 ⟨14 Jan 92⟩ bi-lingual	`format=splain 92.10.27` SLITEX 2.09 ⟨14 Jan 92⟩ tri-lingual
`cpzphysc`	`format=cpzphysc 92.10.15` tri-lingual, (`news zphysc`)	`format=cpzphysc 92.10.16` tri-lingual, (`news zphysc`)	`format=cpzphysc 92.10.27` tri-lingual, (`news zphysc`)
`ppzphysc`	`format=ppzphysc 92.10.15` tri-lingual, (`news zphysc`)	`format=ppzphysc 92.10.16` tri-lingual, (`news zphysc`)	`format=ppzphysc 92.10.27` tri-lingual, (`news zphysc`)
`bibtex`	Version 0.99c	Version 0.99c	Version 0.99c
`makeindex`	Version 2.11	Version 2.11	Version 2.11
`xdvi`	Patchlevel 11	Patchlevel 11	Patchlevel 11
`dvips`	Version 5.495	Version 5.495	Version 5.497

TEX ("big" Version 3.14) and its friends are available on the Apollo-, HP-, and SGI-clusters together with programs for viewing documents on X-stations (`xdvi`) and POSTSCRIPT-printers (`dvips`).

The command `news tex` shows information about the usage and certain features of the installation. It contains pointers to the relevant manual pages: `man tex`, `man latex`, `man slitex`, `man xdvi`, `man dvips` (there is no manual page for `amstex`). See also `man bibtex` and `man makeindex`.

A typical sequence of actions looks like this:

1. **Edit** your input-file "paper.tex" using your favorite editor. (Some editors allow you to execute step 2 from inside the editor.)

2. "`latex paper`" to run LaTeX

3. "`xdvi paper&`" to look at the output on the display (conveniently run as a background task) [if changes are needed, goto 1. -else-]

4. "`dvips [options] paper`"
   to create the PostScript file paper.ps (you may use the "-o"-option to *pipe* the output directly

to the printer, thus avoiding step 5.)

5. "lp[r] [options] paper.ps" to send output to the printer

The standard TEX input files can be found in /usr/local/lib/tex/inputs.

An overview of the currently installed products is given in the table above. "bi-lingual" denotes versions with US-English (\language0) and German (\language1) hyphenation; "tri-lingual" versions have French (\language3) hyphenation in addition. It is intended to adjust all products to the same level soon.

• Using the Emacs Editor

by Makoto Ikeda

Many UNIX machines provide a screen editor, *vi*, as part of the installed operating system. There are a number of advantages in learning this UNIX-default editor, the most obvious being that one can move from one UNIX machine to another and have the confidence of immediately being able to edit and modify text files. However, a more powerful editor, *emacs*, is available from the Free Software Foundation, and can be installed on almost all UNIX systems. It is currently set as the default editor for the DESY HPs and Apollos. The main advantage of emacs is that it can be tailored to suit your taste or a particular environment. The DESY UNIX committee has created standard default settings for emacs so that one will see the same editor on all DESY UNIX machines. Through the HEPIX organisation, emacs should become a standard installation for UNIX throughout the HEP community.

Getting started

The first step is to get the GNU Emacs Reference Card from the user area in front of the computing centre or from the UCO. From the reference card one sees that all commands are accessed through a combination of Control (denoted by "*C-*"), Escape or Meta (denoted by "*M-*"), and regular characters. At first the number of available commands seems bewildering, but actually only a small subset of commands is necessary to get started and for simple editing. Mappings for the various keyboards will also be available soon.
To start emacs, one types

emacs filename

where the filename is usually of the form *xxx.yyy* and *yyy* defines the filetype. If you are on a workstation or an X-terminal, this will open an emacs window, while on a regular terminal or emulator, an emacs session will start on your screen. An editing buffer is created with the same name as the filename;

this name is displayed in reverse video at the bottom of the screen.

The editor is sensitive to the filetype used in the command. For example if you want to edit a TEXfile, then normally the filetype, *.tex*, is used. A predefined mode is then automatically applied to the editing buffer. This changes aspects of entering text that is specific to TEX. There are also modes for programming languages such as C and Fortran, which provide the proper tabbing and indentations. If no filetype is specified or if emacs cannot determine any special characteristics from the filetype, then the default mode with no special behavior is used.

Unlike the vi editor, emacs is always "active", that is to say, text will immediately be entered without any need of pressing a control character first. The arrow keys have been preset for cursor movement and the delete key is used for character rubout. Thus one can create a simple file immediately and the only control characters that one needs to know are *C-x C-c* to exit. Emacs will prompt you on whether you want to keep all changes you have made during your session or not.

The command to suspend emacs, *C-z*, will interrupt the process and push it into the background. In this way, one would only have to start one emacs process per login session. You can then enter and leave the editor without having to reload emacs each time you want to edit a file. The process can be restarted or placed back into the foreground in the usual way through the shell job control commands. The following example illustrates how this is done.

```
prompt> jobs          (finds job number)
[1] + Stopped emacs
prompt> fg %1.        (restarts job 1)
```

Note that on X-terminals, a child process for the emacs window is started automatically, hence the *C-z* command is not necessary and is currently disabled. [*Warning*: Apollo users who login via standard terminals or emulators should not use *C-z*! Currently there is a rather nasty feature in which the emacs session crashes when the process is pushed back into the foreground. HP users should not experience this problem].

Please note that if you entered emacs *without* specifying a filename, then you must save your work first before exiting. Otherwise when you exit, since no filename was originally specified to emacs, all your editing will be lost. Type *C-x C-w* to write to a file. Once a file is written, you may make additional changes in the editing session and any future exit will produce the proper prompt and warning. A list of important commands is given at the end of this article.

The next step

To help in learning more about basic emacs, there is a tutorial that can be accessed by *M-x* and typing *help-with-tutorial*. On the emacs version installed on the Apollo, you can access the tutorial by the shorthand command *C-h t*. The first problem you come across in the tutorial is when you first have to use the control and escape keys. The control key is easily found on standard keyboards, but often the escape key is not in an obvious location. Unfortunately this is one key that is terminal-dependent. The key mapping for your particular keyboard will show where it is. If you are using a non-standard keyboard, then try typing *C-*[or *C-*; which will send the proper ASCII code that an escape key would send. To exit out of the tutorial, type *C-x k*. You will be prompted to kill the tutorial buffer, in which case you just type a carriage return.

In addition to the tutorial there is an online help facility accessed by *M-x help-for-help* or *C-h* for the shorthand command on the Apollo. Since this facility makes use of emacs tools like buffers and windows, it would be helpful to know some important control commands beforehand. If you type an unwanted control character, simply type *C-g* to reset. If the screen gets garbled because of a message broadcast, simply type *C-l* to refresh the screen. Sometimes the help facility may create another window or split the screen. To remove the additional window, type *C-x 1*. The help facility also makes use of an additional buffer to display a help file. You can kill this buffer by typing *C-x k*. This was the same command that was used to kill the buffer used by the tutorial. All these commands are included in the command summary at the end of this article.

Finally for the advanced users who are interested in learning more about emacs, there is an online documentation reader called info. This is accessed by *M-x info* or on the Apollo *C-h i*. This reader allows you to browse through the complete emacs manual (plus other documentation). Note, however, that the reader has its own additional set of commands that are accessed by single regular characters. These commands are summarised in the GNU Emacs Reference Card under the heading Info. First time users of info should go through the primer by typing *h* after they have entered the reader. To leave the reader simply type *q*.

Possible problems with different terminals

All emacs commands are accessible through the use of control and escape characters. Assuming you know how to access these characters, then emacs can be run independent of the terminal configuration. However, to simplify the task of accessing commands, emacs has been configured on each system so that the keypad keys and the supplemental function keys will implement some of the heavily used commands. Therefore it is particularly helpful to obtain a key mapping for your terminal.

The different keyboard configurations rely on the correct setting of the TERM variable. You can check your TERM variable by the command

```
echo $TERM
```

Settings have been created for xterm and the vt terminals. However, it is impossible to cover all terminals and emulators. Usually most terminals can be reset to a vt100. If you are running the Bourne or Korn shell this can be accomplished by

```
export TERM=vt100
```

For the C shell and its variants, use

```
setenv TERM vt100
```

You should now be able to start an emacs session.

A well-known terminal dependent problem is the flow control protocol (scrolling) incorporated into the *C-s* and *C-q* characters on some terminals, terminal servers, and modems. On these devices, these keys are bound to flow control and thus emacs never sees these characters. There are no simple solutions for these devices. Turning off flow control at a terminal will cause other problems such as the loss of characters. Fortunately in the DESY standard configuration, the emacs commands that use these control characters have been bound to supplemental keys. Please see your key mapping to access these commands.

Another related problem is that control settings on some terminal servers, such as Emulex, will also override emacs commands. It is important that you use rarely used control characters (like *C*-\, for example) for your port settings on these terminal servers. Otherwise the control characters that you select will implement the terminal server command rather than the emacs command.

If emacs is not properly configured on your machine, you may get an error message about your DISPLAY variable not being set. Your terminal is not recognized as a non-windowing terminal and thus emacs is attempting to open a X-window. To get around this problem temporarily, type

```
emacs -nw &
```

where the nw (no window) option must be the first option. Please consult your system manager on correcting the emacs configuration.

A quick list of the basic commands

Since the reference card is rather "dense", a listing of the basic commands required to get you editing easily and immediately is given below. Note that the characters used in many of the commands have the first letter of the task you actually want implemented.

Basic cursor and screen movement

Arrow keys	forward/backward and up/down
C-v	view or scroll forward one page
M-v	view or scroll backward one page
M-<	go to top of buffer
M->	go to bottom of buffer

Incremental search: searches as you type text

C-s	search forward
C-r	reverse search or search backward

Type *C-s* or *C-r* again to repeat search

M	escape key alone to exit search

Cutting and pasting

Del-key	kill or delete previous character
C-k	kill or delete to the end of current line
C-@	set mark in preparation to kill a region
C-w	wipe/kill the region from mark to here
C-y	yank/paste previous kill here

Multi-buffer editing

C-x C-f	read a file into an editing buffer
C-x C-s	save your work by updating your file
C-x C-w	write out buffer to a particular file
C-x i	insert a file into current buffer
C-x b	move to another buffer
C-x C-b	list all buffers created so far
C-x k	kill this buffer

Cut-and-paste from buffer to buffer is possible

Emacs commands by explicit calls

M-x help-with-tutorial	access tutorial
M-x help-for-help	help files
M-x info	documentation reader

Error recovery

C-g	abort partially typed command
C-x u	undo last command
C-l	refresh screen

Quitting

C-z	suspend emacs (standard terminals)
C-x C-c	quit emacs

On your own

In addition to the standard functions, other special utilities are available in emacs, including its own mail program, spelling checker, shell command execution, keyboard macros (programming of keys), customization, and even automated psychotherapy (the well-known Doctor program). For more information on these and other utilities, please consult the emacs manual which is available in the UCO. Information about the Free Software Foundation and its philosophy is included in the manual.

Many who are experienced with UNIX and are already familiar with the vi editor, will most likely continue to use the vi editor. However, those who are new to UNIX will find emacs to be a much more powerful editor, easier to use, and they can be assured that it is also widely available on UNIX machines. Versions of emacs are freely available for other operating systems as well including VMS, MS-DOS, OS/2, Atari, and Amiga through anonymous ftp at various internet sites.

• Current Situation of Backup on the Apollo-, HP-, and SGI-clusters

by Peter K. Schilling

Apollo-Cluster

Full backup (once a week) and incremental backup (once a day) to disk for system and user files is done automatically by use of a script (using `cpio` and `compress`) written by the system administrator. (Note: due to lack of disk space, user data backup is not done on the H1 machines). Recovery of user files is only possible with the help of the system administrator (please contact Thomas Finnern) within one day after writing (due to disk space shortage).

Users may backup their data by writing it to streamer or DAT tapes using the standard tar command (`man tar`) on certain machines of the Apollo-Cluster (see `news tar`).

HP-Cluster

Full backup to DAT for selected system data files on the root machine is done automatically once a week by use of the standard software `fbackup`. Recovery on file level may be done by use of the standard software `frecover`. Backup is controlled by management software written by the system administrator.

Users may backup their data by writing it to DAT using the standard tar command (`man tar`). Note: This must be done under personal control of the device on the root machine (Please contact the system administrator Ray Koluvek). User data recovery from DAT is done with tar, too.

SGI-Clusters

A commercial product (*NetWorker*) is being tested in the reconstruction cluster and on the new machine `x4u`. (For details contact Karsten Künne.)

Backup files are automatically written to an *EXABYTE-stacker* under control of a flexible configuration mechanism and may be written individually upon user request. Data can be recovered by users selectively.

Currently full backup is done once a week, incremental backup once a day.

Since *NetWorker* supports clients on a number of different systems, this product may be a good candidate for a general backup facility.

General backup facility

Work is in progress to find a general solution for the backup task.

• Computer Accessible Information: What is Located Where and How to Find it

by Katherine Wipf and Michael Behrens

Does this sound familiar? You have finally filled in all the confusing forms and collected all the necessary signatures from those very, very busy people and now you actually have a USERID and a PASSWORD! You even managed to find a colleague who could spare 5 minutes to show you how to connect to the correct machine and log on. But now your colleague has left and you don't know what to do. You remember hearing that `HELP` is often a useful command, so you type it in. You are somewhat annoyed to find the following incomprehensible question thrown at you:

```
What is the messages number or SCCS
command name?
```

instead of receiving helpful suggestions and answers to your own questions. When you try `INFO` you get a list which claims that a certain `LI1` printer is "down" and informs that `HELP SOCIAL` will bring you up to date with social events at DESY. That's a big help! Didn't someone once say that a "?" sometimes helps? When you try it the computer screen displays:

```
%DCL-W-NOCOMD, no command on line -
reenter with alphabetic first character
```

If this scene is similar to the experiences you have made when trying to get information from a computer, this article may prove useful. In it we have tried to compile a list of what information you can obtain on which DESY computers.

Basic Help for the Available Commands

IBM Mainframe

If you just type in the word `HELP`, you will get a whole screen full of information consisting of uppercase topics on the left and brief descriptions on the right. If you press `<ENTER>` the list continues and you can get full details on any displayed topic by typing an 'H' at the beginning of the corresponding line. There is actually quite a bit of useful information here (especially under `INDEX`, `OVERVIEW`, `COMMANDS`, and `FULL_SCREEN`) if you take the time to look through it. Once you have selected a topic, you can always return to the original `HELP` screen by using the '\' command.

Unfortunately the list of topics contains quite a few terms which are meaningless (even after reading the brief descriptions) to the beginner or the casual user of the IBM. In addition, any number of topics which could be of great interest to a user in need of help (such as "Editing Commands," "Using Tapes," or "Programming") do not appear in the list at all. For this reason it is always a good idea to add a topic you need help with to your `HELP` command. If there is a help text for the topic you chose, you will see a number of lines toward the top of the screen which start with `Select HELP for`. By placing a '*' at the beginning of such a line you can read the corresponding help text. The last `Select` line begins with `Select Keyword` and leads you to a list of all the help texts which contain your topic as a keyword. If there is no help text for your topic you will be shown the keyword list immediately. Sometimes these lists can be very long (`HELP File` produces a list of 69 different helps) and it is difficult to find the information you want. The list can be reduced by specifying several words in your `HELP` command (if you want information about copying files you can enter `HELP File Copy` and get a list with 7 helps).

Central VAX Cluster

On the VAX, typing `HELP` will lead you to a screen which briefly explains the VAX/VMS help facility. If you press `<RETURN>` you will see a list of all the help topics available. Choose a topic by typing at least the first four letters after the `Topic?` prompt, or if you are not quite sure of the name of your command try the topic `HINTS`. Whenever you are reading a help text you can redisplay it (from the top) by entering a '?'.

The VAX/VMS help facility is (unfortunately) arranged in a tree structure. Most help texts have a number of subhelps (usually describing command qualifiers and parameters) which branch out from them and are not accessible without mentioning the parent help. This makes it rather difficult to find certain items of information (for example, if you want to find out how to change your password you must say `$ HELP set password` because `$ HELP password` only tells you how to specify a password for a batch job). Despite this disadvantage, however, most information about VAX/VMS commands can be found without too much trouble in the help facility.

UNIX Machines

UNIX doesn't have a help facility (at least not one that is called help). Giving the command `HELP` on a UNIX machine produces a variety of very unhelpful statements depending on which shell you are using.

Information on UNIX commands is available via the `man` command (short for manual). `man` *name* will produce a full description of the command *name*, provided it happens to be a command and provided a description is available. Unfortunately there is little help available if you simply do not know what the name of a command for performing a certain function under UNIX is (and UNIX command names are in general *not* what you expect them to be). You will not , for example, find `man` entries for names `delete`, `copy` or `logout` on either the Apollo or the HP cluster. A good book on UNIX is probably a better choice for a real beginner.

Finding out Who is Logged on

IBM Mainframe

The command `TSO` displays the number of active users and where they are logged on. You can use the `FIND` or `POINT` commands in this screen to locate specific users (i.e. `POINT userid`). If you only wish to check if a particular user is logged on you can use the command `USER userid`.

In many cases it is possible to find out if a particular user is logged on to a remote EARN/BITNET node by sending a command from the IBM. This information can be very useful if you wish to exchange interactive messages with the user (which can be done with the same IBM command). The syntax of the command is `TELL @node remote_command` where the format of the remote command depends on the operating system of the node you are inquiring about. *Note that the remote command must be typed in upper case letters for IBM machines!* If you want to ask if a specific user is logged on to a remote IBM VM/RSCS system, the `TELL` command would look like this:
`TELL @node CPQ USER userid`
To get the same information from a VAX VMS system you would use this command:
`TELL @node show user userid`
If you try this command out using the node DESYVAX, you will probably be surprised when `TELL` claims that a particular user isn't logged on when you know that he is. The reason for this apparently false information is that only the cluster node VAX58B is actually a BITnet node.

More examples of remote commands and an explanation of how to send interactive messages can be found under `HELP TELL`.

Central VAX Cluster

To see who is logged on to VXDESY you need the `$ SHOW USERS` command. This command has a number of qualifiers and you can give a character string as a parameter if you only want information about userids starting with this string. Among the more useful qualifiers are `/BATCH`, which produces a list of the users currently running batch jobs, and `/NODE=nodename`, which restricts the list to a specified node within a VAX cluster. `$ HELP SHOW USERS Examples` gives a number of examples for this command.

The VAX/VMS facility `PHONE` allows you to check who is logged on to a remote DECNET node, although this is not the primary purpose of the facility. The command `$ PHONE DIRECTORY node::` will give you a list of all the users logged on to the specified node, provided this node has the `PHONE` facility installed. To leave PHONE after looking at the list of users, type in EXIT, or if you want to find out what else PHONE can do, type HELP.

UNIX Machines

The UNIX command **who** will produce a list of the users working on the machine. It has a number of options, such as `who -a` for listing all users in the cluster or `who -q` for a quick (reduced) output listing only the userids and the number logged on. All the available **who** options are described under `man who`.

The command `finger` provides the same information as **who**, but in addition it lists the personal names and information about office location, group and phone number where available. `Finger` can also be used to find out who is logged on to a remote INTERNET node (provided they have a `finger` server and allow remote inquiries). The syntax is `finger @node_name`.

Finding out a User's Group, Phone Number, Home Directory, etc.

IBM Mainframe

The command INQUIRY is provided for finding out a person's full name, userid, group, phone number, and beeper number (if available). You can abbreviate the command to IN and follow it with any string of at least 3 letters and/or numbers in order to obtain a list of all the entries which contain this string (in any part of the entry). For example, the command `IN max` will show a list which contains users whose first name is Max as well as those whose last name or userid contains the string MAX.

Central VAX Cluster

The same information which is available under INQUIRY on the IBM can be accessed on VXDESY via the PBOOK command. PBOOK works exactly like INQUIRY, except that it has a few extra features. The character string does not have a minimum length of 3 and it is also possible to specify more than one string. In this case the list of entries shown is restricted to those that contain <u>all</u> the specified strings.

UNIX Machines

To find our somebody's userid, the full name of a person with an account on a certain system, and sometimes more information, you will use the `finger` command. You have the following possibilities:

`finger name` displays information about a specific user registered on your system.

`finger name@system` shows you the same information for a user or person on another system.

Please note that the access to this information is restricted on several systems, so you will not get full information on all possible systems. Especially requests from remote systems will often not be honoured. The same applies to history information. If you get the reply `Never logged in` for a particular user, this often means only that you have no access to the history file on this machine.

• Phasing out the QMS laser printers

by Peter K. Schilling

Currently six QMS-LG1200 printers exist (destinations L1 ... L6, see: "h dest") and are connected directly to the IBM mainframe. Since the maintenance of the devices is rather expensive, they will be phased out gradually and *replaced* by network printers of the type HP LaserJet IIISi.

The service will cease for the printers L3, L4, and L5 in January, 1993, after the new printers have been installed at those locations. The other three printers L1, L2, and L6 will be replaced by the end of March 1993.

Two of the new printers are already available at the location of L1 (computer center users' area): destinations R02PS2 and R02PS3 (see "h dest"). The new network printers only accept files coded in the page description language "PostScript" (industrial standard).

The NEWLIB "print"-command automatically generates the correct coding depending on the selected printer. It has been enhanced by options to support the new features available with the new printers (e.g. simplex/duplex printing).

Users of GKS-based application programs (as well as GEP and PAW) can generate PostScript-files (and "EPS"-files, see below) for the network-printers. NEWLIB now offers a new "plot"-command for easy plot submission.

The clist GTEX and its variants (like GTEX3) automatically generate the correct coding for the selected destination.

There is one **caveat**, however: Graphics to be included in (La)TeX documents have to be given as "Encapsulated PostScript" (EPS) files when sent to a network printer. Please do "h texmacros" and "SELECT Style-option "psmacro" for including EPS-graphic in LaTeX" and find out about the usage of the "`\psgraphic`" macro for including EPS-graphic.

Note: R2 has prepared a NEWLIB-Clist QUIC2PS for converting old-style "QUIC-stacks" (i.e. a library of pictures generated by the clist PIC2QUIC, see "H PIC2QIC") to a library of Encapsulated PostScript (EPS) members or a set of sequential EPS-files: See "H QUIC2PS".

• A Last-Minute Notice from our Users

Contributed by Marcus Speh

Atavachron – Interacting with Preprint Archives under Emacs

There is a new set of tools available at DESY for interacting with the preprint archives (`hep-lat`, `hep-ph`, `hep-th` etc.) from within the GNU EMACS editor. They offer the user a simple set of commands for getting , storing , processing, and listing papers. These tools are especially comfortable if the preprint contains `PostScript` figures. For more information, read the `news` for `atavachron` on the HP or the Silicon Graphics (SGI) cluster. A complete manual can be obtained in `PostScript` from M. Speh ⟨`marcus@ips102.desy.de`⟩ or read as an Emacs info file (type `info` on the command line).

Private Support for GNU Software at DESY

As a test, some users of free software from the FSF ["Free Software Foundation"] have decided to improve on the support of GNU products. These include the GNU EMACS editor [now available on all `UNIX` clusters as well as on the VAX (beta test)], the GNU GCC compiler for `C` and `C++`, and many others. They usually come with a mature documentation both as TEX output and as on-line manuals (via the GNU EMACS Info reader). Please note that support is **purely private**, coming from knowledgeable users of GNU products: only in rare cases will the group R2, which has lots of other work to do, be able to assist. The sources will be hosted by `x4u` (SGI Workstation), may be `NFS` mounted on other machines, and can be compiled locally.

Volunteers are sought to join the group, especially from other clusters (ZEUS, H1, ... VAX ?) whose users are enjoying GNU products as well. For further information please contact L. Lönnblad ⟨`lonnblad@ips102.desy.de`⟩ or M. Speh ⟨`marcus@ips102.desy.de`⟩.

• Accounting on the IBM

by Michael Behrens

Last November the accounting scheme for the DESY IBM was modified considerably. This article describes the current scheme in detail with some references to the system used before this date.

How the "Kontingent" is derived

The users of the DESY Computer Centre receive a daily quota of CPU time, called "Kontingent". The "Kontingent" is distributed every night, not to single users but to groups of users, e.g. experiments.

The clist `CCUSAGE` on the IBM gives you information how the quota for today was derived, with a detailed explanation via `Help ACCOUNT`.

The "Kontingent" for each group/experiment is computed according to the following scheme:

• Each group gets a fixed fraction of the total available "Kontingent" for the next day, their *default quota*. This default quota is then adjusted depending on the history of this account.

\+ For all jobs which ended on the previous day, the unused part of the time requested is added to the new quota for their group. *(If e.g. a job with medium or high priority requested 6 min and used only 4 min, 2 min will be added to the new quota).*

\+ The unused part of the previous day's "Kontingent" is transferred to the next day within a certain limit.

\- If a group used more than its actual quota the day before, the excess time is subtracted from the new "Kontingent".

\- CPU time under TSO is free up to a fixed amount, independent of the groups part of total "Kontingent". Excess TSO usage will reduce the quota for the next day. EXTRA time (from restricted job classes accounted only at job termination) is added to the previous day's usage as well.

In order to prevent overloading and excessive queueing, an upper limit for the maximum quota for each group is maintained. This maximum is currently 175% of the normal quota; no group may collect more than this limit by the procedure described above.
For special cases a few more rules apply:

- In certain cases a group may have used much more than their quota, but we never distribute a negative quota. If after all modifications the resulting quota of a group would become negative, then it will be zero, possibly for several days,in to compensate for excessive usage.
- Groups with very small "Kontingent" will receive a standard minimum account (larger than their fraction of the total quota) which enables these groups to run jobs with priority at all. However, if they use more than what would be their "Kontingent" without this special rule, they will receive less the day after.

Accounting and Priorities

Jobs may have one of 3 possible priorities: **HIG**, **MED**, or **LOW** (for high, medium and low), depending on the relative importance a user attributes to his job. For details see `Help CLASSES` under NEWLIB. If no quota for HIG or MED priority is left, the priority will be reduced automatically by the system. The HIG time for a group will be 10% of the MED time.

Jobs with LOW priority do not use any "Kontingent", but they will only run if no jobs with medium or high priority are awaiting execution.

In principle every user is free to choose whatever priority s/he prefers, but most groups have introduced policies with regard to job submission and priority selection. Please check what policies exist within your group.

When the "Kontingent" and the excess quota are used up completely, all jobs of that group will get RELPRI=LOW enforced independent of the priority specified.

Interactive sessions (TSO) are accounted as if they had medium priorities.

How the "Kontingent" is consumed

Upon job submission, the appropriate amount is subtracted from the daily quota for all jobs with priority HIG or MED. The time subtracted is the time specified on the JOB-card or the default time of the job class (if the time parameter was omitted).

For jobs with unusually large memory requirements, the time requested is modified by a certain factor, as described below, to compensate for the extra resources used.

Special Rules for large Jobs

The maximum REGION specification for most job classes is currently 8MB. Jobs that require more memory have to use class **G** and specify the REGION request accordingly. For requests exceeding 8MB, the job will be charged 25 % of the specified TIME for each extra MByte requested. What region you have to specify in order to get the memory space you need is explained in `Help REGION`.

Experiences with the new System

In the former system, time requested but not used was simply lost. The distributed time exceeded the true capacity of the IBM to compensate for this loss. Now the unused time is given back and the total quota distributed has been reduced accordingly.

The first experience we made after introducing the new scheme was that we received no comments or complaints at all. This indicates that the transition worked without significant bad effects.

Discussions before implementing the new system indicated that many users would have preferred an immediate pay back upon job completion (or cancellation) over the pay back at midnight. The structure of the accounting system, however, makes this almost impossible to achieve. The availability of an "overdraft" should help, as the time paid back around midnight will compensate for an overdraft during the day.

• Questions and Answers from the UCO

by Katherine Wipf

The User Consulting Office answers quite a variety of questions, but usually only a few people hear the answers. This column tries to bring the answers to more frequently asked questions to a larger audience.

. . . *about VAX/VMS*

∘ *Question:* For quite some time I have been getting the message
`You have 20 new Mail messages`
when I log on, even when I don't have any new mails. Is there any way that I can reset my mail counter?

∘ *Answer:* This mail counter problem is a bug in VMS mail which usually occurs when a user deletes his `MAIL.MAI` file (but can have other causes). Fortunately it is easy to fix. All you need to do is enter VMS mail and give the following command:
`MAIL> read/new`

∘ *Question:* I was editing a file on VXDESY using the EVE editor when the server disconnected my session. I had made quite a lot of changes and would like to know if I can get them back.

∘ *Answer:* While you edit, the EVE editor keeps track of your changes in a journal file. If you leave the editor with the usual `quit` or `exit`, this journal file is deleted, otherwise it is kept to make it possible to recover the changes. To get back your changes from a journal file, call the editor as usual but add the `/RECOVER` option. If you want to recover the changes for `myfile.txt` the command might look like this:
`$ EDIT/TPU myfile.txt /RECOVER`
The editor will then locate the most recent journal file and ask if it should be recovered. Answer with YES and your changes will be restored.

◦ *Question:* For the past two days I've been having problems with the USENET newsreader on VXDESY. When I invoke it with the `NNEWS` command, the system returns the error message:

```
RMS-E-FNF, file not found
RMS-F-IFI, invalid internal file
                     identifier (IFI) value
```

The newsreader always worked before. What is wrong with it now?

◦ *Answer:* The USENET newsreader installed on VXDESY uses a file called `NEWSRC.`, which is stored in your home directory. This file contains information about which articles you have read and which groups you have subscribed to, and NNEWS will not run without it. If the file is missing you will get the error messages you described. For this reason you should add a line to your LOGIN.COM file which will check that you have a `NEWSRC.` file each time you log in and create a new one if you don't have it. The required line looks like this:

```
$ if "''f$mode()'" .eqs. "INTERACTIVE"
   then @desy$root:[newsreader]nnewslogin
```

. . . *about the IBM*

◦ *Question:* Sometimes when I'm editing with NEWLIB on the IBM I enter a `DD` or `CC` line-command and then find that I don't want it (if I've put it in the wrong place or used the wrong command). Is there any way of getting rid of these unwanted line-commands without leaving the current library?

◦ *Answer:* Yes, it's quite simple. By typing the letter X in the first column of the line which was incorrectly marked you can undo the pending line-command. If you have marked a range for copying or moving you can undo the entire range by typeing an X in the first column of the =MSG=> line at the top (or bottom) of the screen. This is the line which says:

```
=MSG=> MOVE range waits for destination
                    -or-
=MSG=> COPY lines wait for destination
```

There is also a simple NEWLIB command that will undo ALL pending line-commands, even if you don't remember where you put them. It is called `RESETL` *(which stands for RESET Line-commands).*

◦ *Question:* I recently typed in the wrong password several times on the IBM, and now I get this message every time I attempt a logon:

```
LOGON REJECTED, RACF TEMPORARILY REVOKING
USER ACCESS
CONTACT YOUR TSO ADMINISTRATOR
```

Could you please reset my password so that I can log on?

◦ *Answer:* The UCO <u>cannot</u> resurrect revoked RACF userids. You must always contact your RACF administrator to have this done. If you do not know who your administrator is, there are several ways to find out:

- Refer to the table on page three of the *DESY Computing Newsletter No.3* (you can get a copy in the UCO)
- Ask someone logged on to the IBM to give the command `HELP RACGRPS`
- Ask someone in your group if they know who the group RACF administrator is
- Ask the UCO to tell you who your RACF administrator is

◦ *Question:* I'm compiling and linking a program in batch mode on the IBM which needs a large number of compiler options. There are so many that they won't all fit on one line. I've tried splitting the CPRM parameter (which indicates the compiler options) onto two lines, but I always get a JCL error. Is there any way of solving my problem?

◦ *Answer:* It is possible to split the CPRM parameter, but unfortunately you must give it a different name in order to do so. The name is PARM.C and you need to use parenthesis to enclose all the compiler options, in addition to apostrophes to enclose all the options placed on one line. This is what PARM.C might look like:

```
//      EXEC    VFORTCL,
//              PARM.C=('OPT(3),VECTOR',
//              'NOSOURCE,DC(COMA,COMB)'),
//                    .  .  .
```

If you happen to have too many linkage editor options to fit on one line, the LPRM parameter can be changed into PARM.L in the same manner. It should be noted, however, that the list of options for PARM (whether PARM.C or PARM.L) is restricted to 100 characters.

∘ *Question:* I got a "dataset warning letter" from the IBM. In this letter I was told that several files had been deleted after previous warnings. I discovered that some of these files were still in my catalogue, and as I wanted to keep one of them, I used DSPROTC to preserve it. Nevertheless it disappeared a few days later.

∘ *Answer:* When this letter was mailed to you, a job with a list of datasets to be deleted had already been created. Since deleting many files for the whole of DESY takes quite some time, you were still able to see your files when your received the letter. Unfortunately using DSPROTC for these files has no effect because it cannot remove them from the list of datasets to be deleted.

If you want to save a dataset that is marked with D in the warning letter, the best thing to do is make a copy of it (you may have to either recall the datasets (*see HELP RECALL*) or retrieve and `fast` them (*see HELP ARCHIVE and HELP FAST*) first).

∘ *Question:* I want to use my own include files for my C program. I never could figure out how to write the file names so that the precompiler can find them.

∘ *Answer:* It is indeed difficult to find out and the manual is not easily comprehensible. Let's assume you want to include a file MYFILE or MYFILE.H in your C program.

You have two alternatives. Assume your userid is `F99ABC`. You could create a sequential file (card image format) named MYFILE.H and then use the following include statement:

`#include "'F99ABC.MYFILE.H'"`

and the compiler will find the file. *Note the full dsname including the userid.*

You might, however, prefer a library which contains all your include files (especially if you have a lot of them). In this case you have to

- Use a library in FB format (or model DCB `CARDIM`) instead of the internal NEWLIB format
- Strip the .H or whatever extension from the name. The member is simply named MYFILE in this case.
- Before compiling, allocate the library under the `DDNAME` USERLIB, for example:
  `ALLOC F(userlib) DA(my.include.library)`

The include statement then will have the form

`#include "MYFILE.H"` or

`#include "MYFILE"`

There are more forms possible. For details see the C manuals.

∘ *Question:* Sometimes I get a notification if new mail arrives for me on the IBM, but often mails arrive without notification so that I only notice them days later when I use READMAIL.

∘ *Answer:* Mail on the IBM arrives over two different channels, either from BITnet/EARN (mail sent to *userid*@dhhdesy3.bitnet) or from Internet (mail sent to *userid*@dsyibm.desy.de).

Currently you only get notified for mail arriving from BITnet/EARN. There are plans to make you aware of incoming Internet mail as well, but the necessary changes are not yet complete. Until then you should use READMAIL regularly to see if new mail has arrived.

. . . *about GEP and PAW*

∘ *Question:* I'd like to be able to print my GEP plots on a postscript printer. Is there an easy way of doing this?

∘ *Answer:* With GEP version 5.3 it is possible to create postscript (PS) files from GEP plots. Note, however, that it can only be done with the GKS version of the new interactive GEP.

Prepare the plot you wish to print and then give the command `Q PFILE`. You will now be prompted for the name of your postscript output file. If you give the name of an existing postscript file, GEP will overwrite it, otherwise GEP will ask if it should create a new file with the name you entered. If you have several plots to print, just repeat the procedure for each one. GEP gives you the option of adding subsequent plots to the postscript output file which you specified for the first plot . Once the postscript file is complete, you can print it with the command `PLOT dest`, where `dest` is any postscript printer (for example `R∅2PS2`).

GEP offers a number of variations for the postscript format, including encapsulated postscript (EPS - for use in LaTeX documents). Here are the options:

`Q PFILE`	—>	PS Portrait Format
`Q PFILEL`	—>	PS Landscape Format
`Q PFILEC`	—>	PS Portrait Color
`Q PFILELC`	—>	PS Landscape Color
`Q EPFILE`	—>	EPS Portrait
`Q EPFILEL`	—>	EPS Landscape
`Q EPFILEC`	—>	EPS Portrait Color
`Q EPFILELC`	—>	EPS Landscape Color

If you want to print your plot without saving it in a file, you can do this with the `Q <dest>` command where `<dest>` is the postscript printer of your choice. This output option will soon be available for the GDDM version of GEP as well.

∘ *Question:* I have some files containing PAW histograms which I created using HRPUT on the DESY IBM. Now I need to manipulate these files with PAW on the computer in my home institute. I've been using RTOA in order to prepare PAW RZ files for transport between VXDESY and my home institute, but it doesn't seem to exist on the IBM. ZFTP doesn't seem to be available either. Is there any way that I can export my files over the network?

∘ *Answer:* Yes, there is a method for converting PAW RZ files into sequential files for file transfer purposes. This is how it works:

1. Start up PAW and open the existing HBOOK file
2. Create an alpha Zebra file (.FZ) with the command:
   `PAW> TOALPHA filename.FZ`
3. Send the FZ file to the remote computer using your preferred method (ftp, mail, copy, or whatever method is available)

Once you've got the file on the right computer:

1. Check the FZ file on the remote computer. Remove any leading blank lines, mail headers etc. Do the same at the bottom.
2. Start PAW on the remote computer and create a new HBOOK file:
   `PAW> HISTOGRAM/FILE 1 filename.RZ ! N`
   " ! N" tells PAW that you want a new file to be created
3. Transfer the alpha information into your new HBOOK file:
   `PAW> FRALPHA filename.FZ`