Proposal for a Common Script and X Environment at DESY and CERN

A. Köhler *- A. Taddei †

April 19, 1994 - Version 1

Abstract

In order to provide a common user environment on UNIX Systems a set of scripts was designed first by the DESY UNIX Commitee (DUX). An updated and with CERN agreed version exists since march 1994. These scripts are covering the interactive environment which includes a shell and X11 environment. The "startup scripts" (shell environment) provide a way for users to login, work in interactive and batch mode for the most common shells and platforms. The other main part is a proposal for a X11 configuration based on the X Display Manager (xdm). Furthermore we consider some configurations (e.g. key mappings) for heavily used programs like the emacs editor.

*DESY Zeuthen - koehler@ifh.de
†CERN - taddei@dxcern.cern.ch

Contents

List of Tables

List of Figures

1 General overview

1.1 Introduction

With the growing number of UNIX workstations from different vendors the task of setting up a common user environment becomes more and more difficult. We felt a need to have a homogeneous environment from a users' point of view in analogy to the HEPVM efforts on the IBM mainframe. The aim of the project was to keep differences between the platforms as small as possible. The challenge is to ensure that all combinations of platforms and software products represent a reliable and consistent user environment.

1.2 The components

To achieve this user environment attention must be paid to different areas.

Shells

The main access to the UNIX operating system operating system is given by the UNIX shell. Any improvement to it can result in considerably more effective use of the system.
There are at least two major flavours, the Bourne shell and the C shell flavour.
Initially the Bourne shell (sh) was developed which provides nearly no comfort to the interactive user. A shell that is better than the Bourne shell for interactive use is the C shell but the good control of input and output as in sh went away. The shell script language was redesigned to resemble the C language. The tcsh as extension to csh has many extra features like command line editing and completion mechanisms.
David Korn from AT&T developed the Korn shell (ksh) and added also many features that made the C shell good for interactive work. The Korn shell became part of System V and even if it is on some systems it had one major problem: it isn't for free.
The Bash shell (Bourne again shell) from the GNU project is a free shell. Like the ksh, bash was based on the Bourne shell language and has many features from the C shell and from other operating systems.
The zsh written by Paul Falstad is also a Bourne-like free shell and like the tcsh has many additional features.

Graphics environment

The graphics environment is based on the X Window System, developed at the Massachusetts Institue of Technology. There are now different releases and versions of X : X11R4, X11R5 and the coming soon X11R6 (April 1994). X is based on the client/server model, in which the application program (the client) does not directly access the display, but communicates with an

intermediary program (the server). The client and server communicate using the X Protocol, which can run on top of UNIX domain sockets, TCP/IP, or DECnet. As X designers chose to articulate "mechanisms not policy", they didn't insist on a particular "look and feel" like OSF/Motif and OPEN LOOK specifications.
Let's be careful about the fact that the server in the X terminology means a display server and not a server in the usual meaning. So one particular client is the window manager that allows you to move, resize, and iconify windows. But of course it has no responsibilities about the existence of windows themselves which are ruled by the X server and it can be local or remote. There are different startup methods but from our point of view the X Display Manager (xdm) is the best method for starting X sessions. This method will be discussed in section 5.

Software products

As the user needs tools to work, those tools need customization and adaptations. There are interactions between software control files and login scripts. Key bindings and key mappings are examples of control files that should be set carefully and provided for groups of users. So, editors, mail readers, news readers and browsers are examples of classes of software where control files should be maintained and supported for the average user. This list is not complete and can be enlarged. In the following some classical cases will be discussed (emacs, etc.)

2 Shell environment

2.1 Expectations from standardized scripts

With the design of a shell environment for the user you have to consider different views.

User perspective

The user wants a guaranteed functionality so that on all platforms the same feature works in the same way. Especially for inexperienced users a standardized look and feel is very important. Having a set of variables that describe the system the user can test these variables in his own scripts and becomes independent from some system dependencies.

System administrator view

He wants host-independent scripts which he can install on all systems with a minimal amount of modifications or configuration. The number of files to maintain should be kept as small as possible. In the scheme of standardized scripts the system administrator must have the chance to make local adaptations corresponding to individual needs.

The shell environment comes into play in three different situations

- at login-time some scripts are run once at login or if you start a login shell
- interactive invocation for each new interactive shell. There you can configure some features of the shells that modify the interaction with users(eg. key bindings, completion mechanism)
- non interactive invocation for each subshell. Here you should avoid tty related commands or commands requiring tty input

A proper user environment also has to be guaranteed after changing to different shells, in subshells, after the su command and for the remote commands like rsh and in the batch environment.

The scripts should support all major shells used on Unix, the Bourne flavour shells (ksh,bash and zsh) and C flavour shells (csh,tcsh).

In order to understand the layout of our proposed scripts it is necessary to understand which files are executed by default by a given shell (table 2.1).
For csh and tcsh there are additional system dependencies. Also the names of the scripts may differ not mentioned in the table. On some UNIX systems in csh global initialisation files are executed while the default action in tcsh on some systems is not to execute global scripts (e.g. IRIX).

Table 1: Standard configuration files for the different shells

Shell	Conditions	Executed scripts
csh	always	$HOME/.cshrc
	login	$HOME/.login
tcsh	always	/etc/csh.cshrc
	login	/etc/csh.login or $HOME/.login
	interactive	$HOME/.tcshrc or $HOME/.cshrc
ksh	always	ENV variable if set
	login	/etc/profile or $HOME/.profile
	interactive	$HOME/.kshrc
zsh	always	/etc/zshenv
		$HOME/.zshenv
	login	/etc/zprofile
		$HOME/.zprofile
		$HOME/.zlogin
	interactive	/etc/zshrc
		$HOME/.zshrc
bash	always	not possible
	login	/etc/profile
		$HOME/.bash_profile or $HOME/.profile
	interactive	$HOME/.bashrc
	non-interactive	ENV variable if set

2.2 HEP startup files

2.2.1 Description of the components

We have subdivided the scripts in several files according to the schemes used by tcsh and zsh because these shells provide the greatest flexibility to assign scripts to the different cases mentioned above.

Standard HEP configuration files

Table 2: Configuration files for HEP environment.

Conditions	Bourne shell flavor	C shell flavor
always	/etc/zshenv	/etc/cshrc
login	/etc/zprofile	/etc/csh.login
interactive	ksh: /etc/kshrc bash:/etc/bashrc zsh: /etc/zshrc	csh: /etc/csh.cshrc tcsh:/etc/tcshrc

For ksh users you have to link /etc/zprofile to /etc/profile.

These files are standard files and should not be changed. On some systems the ksh and csh don't have any global initialization files. In such a case you have to make sure that the /etc/profile or the /etc/csh.login will be sourced by each user profile. For tcsh you have to consult the man page, which files are sourced. If your system is explicitly mentioned in the section on init files, you probably have to modify the tcsh source to get global initialization files.

All files for local adaptations are contained in the directory /etc/local[1]. There is one configuration file for the setting of environment variables called /etc/local/sys.conf. It will be parsed both from the C shell like and Bourne shell like scripts. In the following we give brief explanations for the meaning of environment variables.

- **OS** is the name of the operating system obtained usually with uname
  - HAS TO BE SET -
- **ROOTPATH** This path is the search path for the root user.
  - HAS TO BE SET -
- **USERPATH** This path is the search path for a normal user.
  - HAS TO BE SET -
- **BINTYPE** setting should reflect the binary compatibility for executables. A supplied script called bintype will evaluate the bintype automatically. You can specify different levels for the evaluation.
- **RUBOUT** defines the RUBOUT sequence which is defined for your keyboard. It can be set to BackSpace or Delete and can be modified by the user with a file $HOME/.BackSpace or $HOME/.Delete.
  On DEC VT terminals, the del (delete) key has always been used as a rubout key, i.e., to delete the previous character. The backspace key on VT terminals is designed to move the cursor back one space on the line WITHOUT deleting the previous character. (A DEC VT terminal does NOT have a key specifically designed to delete the character directly under the cursor.)
  On IBM PC keyboards backspace is used as rubout key, not the delete key. This left the word "delete" to be used for the "delete the character under the cursor" function on the keypad.
- **X11BIN** defines the directories where you can find the X11 binaries
- **MAIL** is set to the full path of your incoming mailbox file.

[1] This is the default and should not be changed

- **MANPATH** lists where you can find manual pages used by commands like man and xman
- **SYSTYPE** reflects the flavor of the UNIX system:
  sys5.3 for system V
  bsd4.3 for BSD UNIX systems
- **YP** reflects the status of NIS (Yellow Pages) and is used for defining an alias passwd to yppasswd
  known values: YES or NO

Other adaptation files are:

- /etc/local/env.[c]sh adaptations for the part which should be always executed
- /etc/local/login.[c]sh adaptations for the login part
- /etc/local/aliases.[c]sh additional aliases/functions
- /etc/local/termset.[c]sh file for term setting corrections

Some examples for these files will be given later in this document.

2.2.2 Description of dynamic behaviour

In this section we describe some special features of the environment, login and interactive parts of the scripts.

Environment which is always executed

The following actions are performed

```
check for login permissions (/etc/nologin)
if (! HEP_ENV) {
    /* Basic shell environment */
    setting of path
    setting of environment variables (/etc/local/sys.conf)
    setting of other environment variables (GROUP, ENV, ..)
    local adaptations (/etc/local/env.[c]sh)
               }
if (Prompt) {
    setting of prompt
```

```
    sourcing of the interactive parts
              - defining aliases
              - enable advanced features
             }
```

If there exists a /etc/nologin file the user cannot login and he gets a message about the reason (standard mechanism on many systems).
For improving the execution effiency an environment variable HEP_ENV is defined. If this variable is set then the environment is assumed to be partly set and in subshells only the interactive part is executed again.

To manage a change of a shell to the ksh or bash the ENV variable is set to /etc/zshenv.

This environment part includes also the setting of the following environment variables which the user can use in his own scripts:

- **USER** is defined because some systems only set the variable LOGNAME
- **GROUP** is set and can be used for a group support
- **PATH** will be extended as mentioned in the sys.conf section
- **HOST** is set to the hostname (without domainname)

Login part

The following actions are performed

```
source the environment file if it was not executed yet (for ksh,bash,csh)
set ENVIRONMENT variable
if ( ! BATCH ) {
  terminal type check and changing to a proper type (/etc/local/termset.[c]sh)
  set terminal characteristics
  check the incoming folder for mail
  local login scripts /etc/local/login.[c]sh
                }
```

Because in most cases the csh, ksh and bash do not have any global configuration files which are always executed the actions described before are performed in addition if necessary.

In order to avoid tty related commands or commands requiring tty input a check for batch mode takes place. In the Bourne-like shells you are in interactive mode if a file descriptor is associated with a terminal device, in the C-like shells you are in interactive mode if the prompt is set. Under some circumstances, e.g. if you login via a terminal server then it is possible that you get an improper terminal type (eg. dumb, network). An optional file /etc/local/termset.[c]sh changes such types to a terminal type which is described in /usr/lib/terminfo or /etc/termcap.

Afterwards some settings of the terminal characteristics will be made (for instance defining the erase key dependent on the RUBOUT variable).

Other stty settings cover the interrupt key (Ctrl C) and the suspend key (Ctrl Z).
After the execution of the login file for local adaptations (/etc/local/login.[c]sh) a check on incoming mail is done.

Interactive part

The last part is used for settings of the interactive environment.
The prompt acts as a reminder of where you logged into and shows your current directory. Where possible the tilde notation for home directories is used. In all cases the superuser is indicated by a hash sign ("#") in his prompt.
A small set of common aliases or functions is defined. You can use these aliases in all shells and systems in the same way.

- **pp <pattern>** is an alias for a grep in the process list defined to hide the system differences between System V (ps -ef) and BSD Systems (ps -aux).
- **ll** for a long directory listing which is defined on many systems as a default command.
- **ff <pattern>** for a frequently-used version of the find command. The find alias searches from the current directory.
- **nman <filename>** for manual pages which are not located in the manpath (nroff -man <filename> | $PAGER)
- **passwd** for systems where the environment variable YP is set. to "YES" . The yppasswd is aliased to passwd.
- **rs** as alias for eval ‘resize‘

The following special interactive features of the shells will be enabled if possible:

- History mechanism
  The history file is set to $HOME/.history.$HOST.

- Command line editing with the arrow keys (doesn't work in most csh versions)
  up arrow up-history
  down arrow down-history
  right arrow forward-char
  left arrow backward-char
  In the zsh and tcsh you have also key bindings like the emacs key binding described later, especially for a history-search-backward and history-search-forward with the Page-Up/Prev-Page and Page-Down/Next-Page key and the Insert and Delete key, if present, work as expected.

- All described shells support the word completion feature. In typing commands only a unique abbreviation is necessary. When you type a TAB (zsh,tcsh,bash), ESC (csh) or ESC ESC (ksh) it will complete the name for you. For the tcsh and zsh a context dependent completion mechanism for the most frequently used commands has been configured. For example a cd command followed by the TAB key only completes (or displays, if ambiguous) directories.

- In some shells directories are indicated with a trailing '/', executable files with a '*', symbolic links with a '@' and FIFOs (named pipes) with a '|'.

For more special features see the manuals of these shells.

3 System administrator tasks

3.1 Installation

<u>**Test Installation**</u>

You can install the startup files in your home directory for tests with the following steps[2].

- 1. unpack the tar file and set the ETC variable in the Makefile

- 2. run make test-all[3]
  This appends some lines in your dot files, so that initialization scripts will be executed as explained above.

- 3. prepare the sys.conf file (see 3.2) and make your adaptations in the local directory $ETC/HEP/etc/local

- 4. now you can login

[2] Be careful not to do it while in account root
[3] Or test-zsh, test-csh, etc.

Real Installation

If you want to install the scripts as a global environment you have to do the following steps as the root user:

- 1. make sure that you have made all local adaptations in the local directory and you have prepared the
  sys.conf file
- 2. run make install-all

There is also the possibility to install these scripts only for a specific shell with:

make install-zsh   install the zsh profiles

make install-sh   install the ksh/bash profiles

make install-csh   install the [t]csh profiles

The install command extracts the real files from your test HEP files, saves the original profiles from /etc to /etc/filename.saveHEP and copies all the new startup files to /etc and /etc/local.

Look into the ksh man page if your ksh has a global initialization file. For sh and ksh without a global initialization file make sure that the /etc/profile will be sourced by each users .profile.

Please consult the csh man page for the system-wide startup files. In most cases the names are /etc/csh.cshrc and /etc/csh.login.

Known exceptions:

ConvexOS : /etc/cshrc and /etc/login

IRIX : /etc/cshrc and /etc/csh.login

On these systems you can link the new /etc/csh.cshrc to /etc/cshrc and the new /etc/csh.login to /etc/login.

For systems which do not have system-wide csh startup files the files /etc/csh.cshrc and /etc/csh.login have to be sourced from the users' .cshrc and .login file.

Known problems:

The make script uses the fact that the environment variables (e.g. $HOME) are imported from the shell environment. On some systems (e.g. ConvexOS) this feature of make is not supported, therefore in such cases GNU-make has to be used instead.

3.2 Modifications by the system administrator

In this section we explain possible modifications by the system administrator. The system administrator can customize these scripts according to his needs.

Setting of environment variables

For the setting of environment variables you should only use the sys.conf file or if this does not exist the sys.conf.$OS file. So you have always an overview about the variables you set and there is only one location for Bourne and C shell flavours.

The syntax is very simple:

<environment_variable>=<value>

Note: If you use variables in this file you have to quote them. After the variable definition additional spaces on the line are not allowed.

In order to define a position in the PATH for a variable used later you can specify this in the following way
USERPATH="/bin /usr/local/bin "\\\$GROUPPATH""
The GROUPPATH variable can be set later for instance in a group file.
setenv GROUPPATH /group/bin
set path=(‘eval echo $path‘)

/etc/local/env.[c]sh

In the environment files /etc/local/env.[c]sh you can make any modifications to your environment, but be careful with commands producing output to the standard output or that set terminal characteristics. This output prevents the usage of remote commands like rcp.

/etc/local/login.[c]sh

The login files /etc/local/login.[c]sh are provided for the login part. Here you can modify terminal settings with stty.
If you have problems with your current window size after a login via telnet you can use the resize command to set TERMCAP capabilities to the current window size.

Example for csh:

```
if ( $term == "vt100" || $term == "xterm" || $term == "xterms" ) then
      if ( $OS == "SunOS" ) then
             set noglob;/bin/sh -c "eval `resize`"
      else
             set noglob;eval `resize -c`
      endif
endif
```

Example for sh:

```
if [ "$TERM" = "vt100" -o "$TERM" = "xterm" -o "$TERM" = "xterms" ]; then
        eval `resize -u`
fi
```

/etc/local/aliases.[c]sh

The files /etc/local/aliases.[c]sh can be used to add some aliases or functions in addition to the standard set of aliases and functions. It is recommended however to keep additional definitions at a bare minimum in order to stay compatible with other sites. There should ideally be a repository that keeps track of all additional definitions that HEP sites have made in order to avoid name clashes and different namings of the same command sequences.

/etc/local/termset.[c]sh

The files /etc/local/termset.[c]sh ensure that the term variable has a reasonable value and redefine it otherwise. This is not a standard file but it has proved to be useful to have in all cases a term variable that matches entries in the terminfo database or in the termcap file.

3.3 Group support

In the proposed scheme you have also the possibility for group support. For the group support you can use the GROUP variable returned by the default scripts. To define a group-dependent environment in the general case you will have to create four files per group. These are the /etc/local/$GROUP.env.[c]sh where you can customize special group environment variables or aliases and the /etc/local/$GROUP.login.[c]sh used for terminal settings or special login messages. These files will be sourced by the /etc/local/env.[c]sh and /etc/local/login.[c]sh.

Examples:

env.csh:	source /etc/local/$GROUP.env.csh
env.sh:	. /etc/local/$GROUP.env.sh
login.csh:	source /etc/local/$GROUP.login.csh
login.sh:	. /etc/local/$GROUP.login.sh

In complex environments with a group file shared by many machines it can be useful to provide an indirect sourcing of these groupfiles so that the /etc/local/$GROUP.env.[c]sh files contain only the source command for the real group files. The group login can be sourced in the same way.

Examples:

```
               source                               source
env.[c]sh    --------->  $GROUP.env.[c]sh      --------->  group env file

               source                               source
login.[c]sh  --------->  $GROUP.login.[c]sh    --------->  group login file
```

4 User customizations

The user can customize his environment with his own dot files.
We will give some examples for tailoring the user environment to personal taste.

Table 3: Standard users dot-files.

Shell	Conditions	Executed dot files
csh	always login	.cshrc .login
tcsh	always login	.tcshrc or .cshrc .login
ksh	login interactive	.profile .kshrc
zsh	always login interactive	.zshenv .zprofile and .zlogin .zshrc
bash	login	.bash_profile or .profile

We will give some examples for tailoring the user environment to personal taste.

How can I change the behaviour of the RUBOUT sequence?

The user can modify it to 2 values : BackSpace or Delete.
Backspace : the user needs to create a $HOME/.BackSpace file.
Delete : the user needs to create a $HOME/.Delete file.

Table 4: How can I change my prompt?

Shell	File	example	Result
csh	.cshrc	`set prompt='\!>'`	`24 >`
tcsh	.tcshrc or .cshrc	`set prompt='\! %~ %#'`	`24 ~`
ksh	.kshrc	`export PS1='\! >$'`	`24 >`
zsh	.zshrc	`export PS1="%S\[%T\]%s"%~ %# "`	`[23:55] ~ >`
bash	.bashrc	`export PS1='\${PWD} \! >'`	`/bin 24 >`

If you want to include the current path in your C-Shell prompt then you have to define the following
alias for the cd command.

```
alias cd chdir \!:\* \; set prompt='${cwd}\[!\]\ ' \; setenv CWD '$cwd'
```

Table 5: How can I define my own aliases or functions?

Shell	File	example
csh	.cshrc	alias rs 'set noglob;eval `resize`;unset noglob'
tcsh	.tcshrc or .cshrc	alias H 'history -r | fgrep "\!*"'
ksh	.kshrc	alias 'mr=chmod +r'
zsh	.zshrc	alias 'rs=eval `resize -u`'
bash	.bashrc	mx() {chmod +x ${*}}

Table 6: How can I append an additional path?

Shell	File	example
csh	.cshrc	set path=($path $HOME/bin)
tcsh	.tcshrc or .cshrc	set path=($path $HOME/bin.$OS)
ksh	.profile	export PATH="$HOME/bin:$PATH"
zsh	.zprofile	export PATH="$HOME/bin.$OS:$PATH"
bash	.bash_profile or	
	.profile	export PATH="$HOME/bin:$PATH"

Table 7: How can I set my own environment variables?

Shell	File	example
csh	.cshrc	setenv EDITOR vi
tcsh	.tcshrc or .cshrc	setenv PRINTER ps_rz1
ksh	.profile	export VISUAL=emacs
zsh	.zprofile	export ORACLE_SID=v712
bash	.bash_profile or	
	.profile	export PAGER=less

5 Graphics environment

5.1 Introduction

The X Display Manager, xdm, provides a way for users to log on and start initial clients. For a user who logs in with an xdm login box the name and the password are authenticated using the same mechanism as the login program. Instead of running an interactive shell, xdm runs a series of shell scripts. When users finish their X sessions, xdm resets each display for the next user.

Xdm also provides a way for administrators to configure environments side-wide. It can be configured to control logins on multiple X Servers connected to the same machine, creating customized sessions, and offering some basic network security features.

Xdm was introduced with X11R3 in 198?. The XDMCP, introduced in X11R4, is a protocol shared by the xdm client and X servers throughout the network. Using XDMCP, the X server has the responsibility of actively requesting an xdm connection from the host.

Xdm can be configured through a set of ASCII files

xdm-config	Resources specified for xdm.
Xservers	A list of servers to be explicitly managed by xdm.
Xsession	The initial startup script used by each individua X session.
Xresources	Resources to be loaded via xrdb by servers managed by xdm.
Xaccess	A file for configuring access control for XDMCP (new in X11R5).
GiveConsole	A shell script that changes the ownership of the console to the user (new in X11R5).
TakeConsole	A shell script that changes the ownership of the console back to root (new in X11R5).
Xsetup_0	A shell script used for display setup specific to the local console server (new in X11R5).

In addition to these standard configuration files there were created some files to customize the X environment in a heterogenous environment to support different keyboards and to provide X connectivity.

5.2 System administration files

In addition to the standard configuration files as mentioned above there will be provided some special files:

- xsession.env
  This file is needed for setting environment variables used by the Xsession script.

XDMPATH	Variable to find xdm configuration files (standard is /usr/lib/X11)
BINTYPE_SETUP	Variable to set the script to run architecture-specific programs
HOST_SETUP	Variable to set the script to run host-specific programs
RESOURCES	Variable to set the X resource file for heavily-used clients (e.g. xterm). Resources will be loaded to the X server resource database with xrdb.
SESSIONLOG	Variable to switch on/off the logging of the session (date, user and display) in a file
SESSIONLOGFILE	Variable to set the session logfile.
FAILSAFE_XTERM	Xterm which runs in failsafe session (e.g. xterm, hpterm, aixterm)
LOGINWINDOW	Xterm which runs in normal session with a login shell and creates a utmp entry
USER_SESSION	Variable to set the script for the user Xsession (standard: $HOME/.Xsession)
USER_RESOURCES	Variable to set the user X resource file which will be merged into the X server resource database
WM	Variable to set the window manager
XTRAS	Variable to set the user specification file for the additional X clients to load (standard:$HOME/.Xtras)

- Xsession.$BINTYPE Script to run architecture-specific programs specified by variable $BINTYPE_SETUP

- Xsession.$HOST Script to run host-specific programs specified by the variable $HOST_SETUP

- sys.Xresources X resource file to set resources for heavily-used clients

- xsession.log Logfile if the variable $SESSIONLOG is switched on

The Xsession will be splitted according to the login shell flavour so that the global configuration files :

C-shell flavour	/etc/csh.login
Bourne shell flavour	/etc/zprofile or /etc/profile

and individual files :

[t]csh	$HOME/.login
zsh	$HOME/.zprofile are executed.
ksh,bash	$HOME/.profile

The last client in the Xsession is the window manager specified with the WM variable. If the variable is not set or the specified window manager is not found, the Motif Window Manager as standard window manager will be executed. You can also specify a file to load the local

window manager, for instance for X-Terminals.
The session ends if the user selects the logout button in the root window manager menu.

5.3 User-specific files

The user can specify the following Xsession variables with a file called $HOME/.xsession.env. A short example will follow after the explanation of the variables.

USER_RESOURCES Variable to set the user X resource file which will be merged into the X server resource database (standard: $HOME/.Xresources)

LOGINWINDOW Xterm which runs in the normal session with a login shell and creates an utmp entry

WM Variable to set the window manager

XTRAS Variable to set the user specification file for the additional X clients to load (standard:$HOME/.Xtras) Here is a short example file.

```
if [ $BINTYPE = "SunOS" ]; then
            /usr/local/bin/X11/xclock &
            /usr/local/bin/X11/xsetroot -solid green &
            /usr/local/bin/X11/xterm &
         else
            /usr/bin/X11/xclock &
            /usr/bin/X11/xsetroot -solid blue &
            /usr/bin/X11/xterm &
         fi
```

USER_SESSION Variable to set the script for the user Xsession (standard: $HOME/.Xsession) This user-specific startup script will be executed by the system-wide Xsession script after the loading of the Xresources, the keyboard handling and the login procedure which set the login environment. The user has to ensure that the last X client he starts runs in the foreground.

Short example for $HOME/.xsession.env file:

```
XTRAS=$HOME/.Xclients
if [ $BINTYPE = "SunOS" ]; then
    WM=/usr/openwin/bin/olwm
else
    WM=/usr/bin/X11/mwm
fi
```

A summary of the dynamic behaviour of how the system sets the X environment through the control files is given in figure 1.

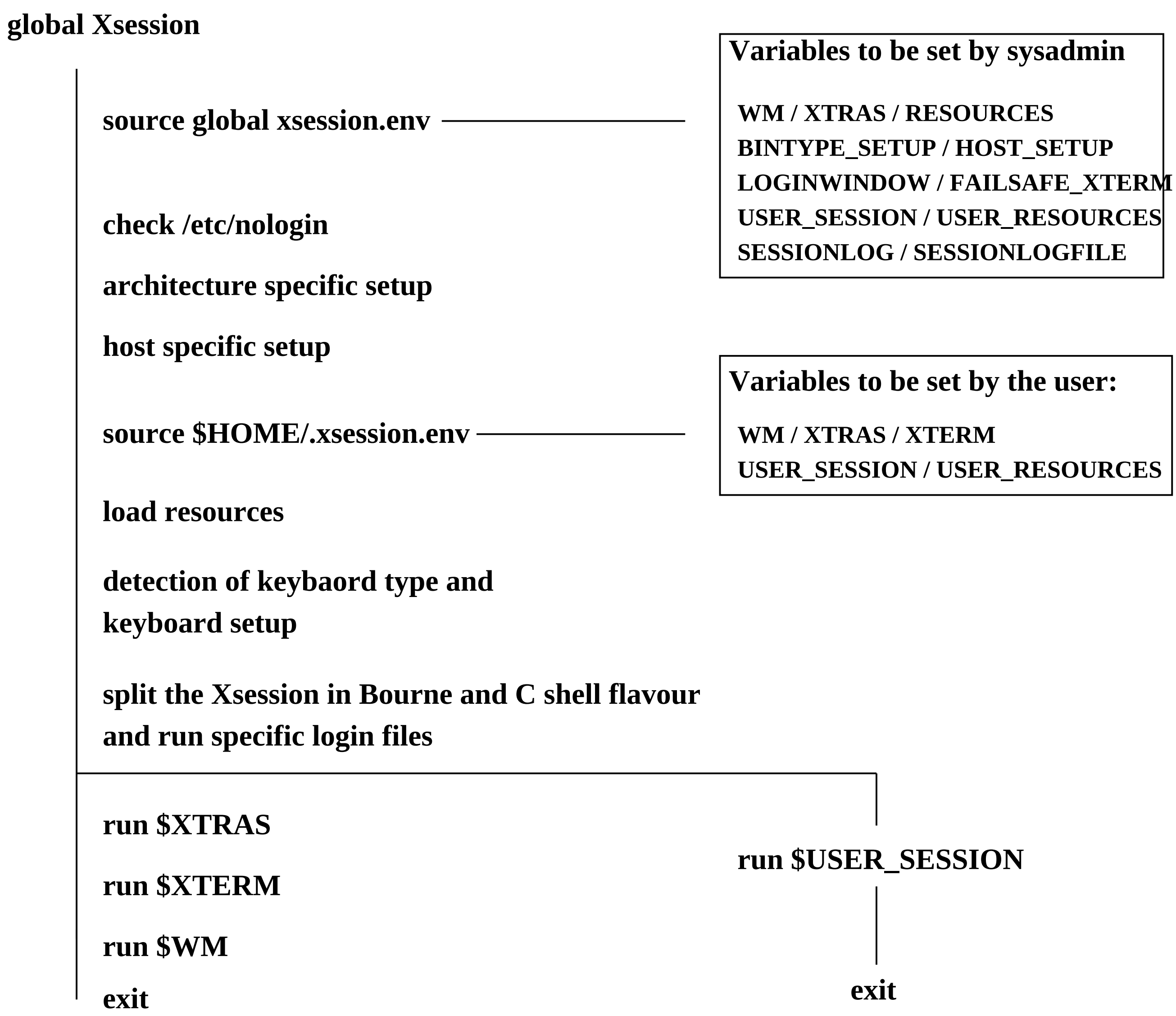


Figure 1: Dynamic behaviour of the X environment

5.4 Keyboard support

Currently there is no way to detect the keyboard type automatically. In order to provide a keyboard support we will provide a tool called **special.keyboard** which is called from the global **Xsession** file. The file looks for a map file **.keyboard** in the user home directory or a global file **special.keyboard_list** in bf $XDMPATH/options. If these files do not exist or there is no entry for the specific displays then this tool queries the vendor of the X server with an xrdb -symbols and will assume you have an standard keyboard from this vendor.
An additional tool xguesskbd which uses the public domain program xkeycaps can create an

entry in the map files. The script starts the xkeycaps program with the guessed keyboard. If this is not the correct keyboard layout the user can select another keyboard. After finishing the xkeycaps program he will get a list of supported keyboard types and can set this entry by hand otherwise the guessed keyboard type will be set automatically in the map files.

The keyboard names in Appendix A should be used.

The map file should have the following syntax:

Xterminalname:keyboardname
eg.:
xtcnsw07.cern.ch:NCD101

For each supported keyboard the system administrator can create a
file $XDMPATH/options/special.$KEYBOARD where he can run commands like xmodmap or xrdb to customize the specific keyboard. If the user wants to customize his terminal he has to create a file $HOME/.keyboard.$KEYBOARD.
An example for such a file looks as follows:

```
#!/bin/ksh
# Define 'Compose' key as Alt key
/usr/bin/X11/xmodmap - << !
clear   Mod1
keycode 33      =      Meta_L
add     Mod1    =      Meta_L
keycode 110     = BackSpace Delete
!
/usr/bin/X11/xrdb -merge <<!
vaxterm.vt100.Translations: #override \
        <Key>BackSpace:         string(0x7f) \n\
!
```

5.5 Tools for X connectivity

In order to start X clients on remote hosts it is necessary to transfer the DISPLAY environment variable. There are two recommended tools to do this.

- **xrsh** starts an X11 client on another host
  Usage: xrsh [-l username] [-auth authtype] [-screen #] [-pass envlist] [-debug] [-debug2] hostname [X-command [arguments...]]

This tool runs the given command on another host with $DISPLAY set in the environment and handles authentication based on the value of $XRSH_AUTH_TYPE or the value of the -auth command line switch.[4]

- **xrlogin** executes a rlogin call with the transfer of the DISPLAY variable. This variable will be packed together with the TERM variable and has to be unpacked on the remote site. The HEP startup login scripts support this unpacking.

There are also tools with a special key mapping.

- **vaxterm** start an xterm with a VT mapping.
  For some special vms editors the german language is supported in the following way:

```
     <Alt>a     maps to a "dotted" a
Shift<Alt>a     maps to a "dotted" A
     <Alt>o     maps to a "dotted" o
Shift<Alt>o     maps to a "dotted" O
     <Alt>u     maps to a "dotted" u
Shift<Alt>u     maps to a "dotted" U
    <Alt> s     maps to a "sharp" S
```

  Before this mode can be used a user must type :
  `set terminal/eight`
  on the vms system.

- **x3270** opens a telnet connection to an IBM host in an X window.

5.6 Window Manager Configuration

In order to have a common behaviour the window manager should be configured in a standardized way.

Button Bindings

It is useful to customize a set of button bindings that are used to configure window manager behaviour. A window manager function can be assigned to situations when a button press occurs with the pointer positioned over a framed client window, an icon, or the root window. A Motif Window Manager configuration can be found in Appendix B.

[4] See the X-Security section

5.7 X-Security

X runs in a networked environment and because of the X11 design, the workstation you are working on can be accessed by any other host on the network. So the possibilities for abuse are considerable. If your environment is not protected in the proper way, an intruder can "xmelt" your screen, send you funny (or not) pictures via "xsetroot" or spy what you are doing, capture your passwords or take control of your mouse or your keyboard.

We have to distinguish between host and user-based access control mechanism. The xhost client can be used to give (or deny) remote systems access to the X-Server interactively. But other people working on this remote system can also access your display.
A far better way is to use the user-based access control introduced with X11 Release 4. The most common method of user-based access control is the mechanism known as MIT-MAGIC-COOKIE. X11 Release 5 provides two new schemes: XDM-AUTHORIZATION-1 and SUN-DES-1.
Another way for display access control uses the public domain tool mxconns. It creates a virtual X server running on the specified display and prompts the user for each new connection, asking whether the connection should be accepted or not. All those systems are studied in [3] and [1].

6 Configuration of software tools

Keymappings

The aim of this key mapping is to describe a way to tailor the interface between the user and the operating system for widely used keyboard layouts (VTxxx-Style,IBM/PC-Style) to inspire the creation of a common interface that is used at more than one site. The idea behind the adaptation of the programs to the keyboards is to help the unexperienced user in performing tasks and to minimize the number of keystrokes. Ideally the tailored system should be self-teaching, for instance there should be at least a common help key (Help,F1,PF1) with a context sensitive help.

emacs

The most important tool the user has to deal with is the editor. A de-facto standard editor is the emacs editor. For this editor we have defined a key mapping for a generic keyboard.

Keyboard mapping principles:

- What you see on the keys is what you get. The right hand numeric keypad should be mapped to VTxxx style
- Fn keys for global functions: F1 help - F10 save
- Missing function keys F13 - F20 will be available as shift F3-F10 if possible
- Numeric keypad for local/edit functions
- Special keys on extended keyboards should be mapped to appropriate functions in the application programs or left unmapped if the suggested function is not available
- Remove/Delete key should delete the character under the cursor
- Backspace key should delete the character to the left of the cursor
- Avoid combinations (ESC-,Ctrl-,Alt-), if possible

The use of the function keys depends on whether you are running inside a terminal emulator (xterm), or whether you are allowing emacs to create its own window.
If you are running emacs inside a terminal emulator window, then it behaves exactly as it does on any other tty. In this case , for function keys to be useful, they must generate character sequences that are sent to the program running inside the window. If emacs manages its own X-window, it receives "KeyPress" events from the X-Server. These KeyPress events contain a "X-Keysym" Code. You can change this keysym Code with the xmodmap program.

In emacs version 19 you can bind function keys like this

```
(define-key global-map [f1]    'help-for-help-with-keypad-map).
```

But this works only under X windows. If you are running the emacs inside a terminal emulator you have to find out what character sequence is generated by the keystroke sequence you wish to bind to a command. We assume that the terminal emulator returns VT sequences.

For the installation you have to make sure that the files keys.el and keys-nw.el will be loaded. You can append the load-keys.el to your $HOME/.emacs file or you can append this file in the system-wide initialization file /usr/local/lib/emacs/site-lisp/site.start.el which is loaded before the $HOME/.emacs file.

The load-keys.el file adds the directory $HOME/elisp in front of the search path for the file to be loaded and contains the loading of the key mapping files if you run emacs version 19.xx . Some additional useful functions will be defined.

- help-for-help-with-keypad-map	Shows the key mapping for emacs (F1/PF1 or help key)
- my-search-forward	Saves search string for repetition and do search-forward
- my-search-backward	Saves search string for repetition and do search-backward

The proposed key mapping is shown in appendix C and D.

newsreaders

One variable which should be set is the NNTPSERVER which indicates which server should be contacted for news reading, posting, etc. This variable is used by all known newsreaders.

For example, one could include of the following line

```
setenv NNTPSERVER news.cern.ch
```

in the file /etc/local/env.sh.

Mosaic

One example of a variable which can be set for Mosaic is the WWW_HOME variable. This variable could be made group dependent for example.

```
setenv WWW_HOME http://www1.cern.ch
```

The Browser less

You may define your own less commands by using the program lesskey to create a file called ".less" . This file specifies a set of command keys and an action associated with each key. The key bindings for less can be loaded with the -k option.

```
setenv LESS "-e -i -M -w -k/usr/local/lib/less/.less"
```

For the explanation of the other options consult the man page for less.

7 How can I get the files?

In order to maintain single source and permit easy porting changes for individual sites we have decided to use the code management system CVS. It works by holding a central 'repository' of the most recent version of the files. You may create at any time a personal copy of these files by 'checking out' the files from the repository into one of your directories. If at a later date newer versions of the files are put in the repository, you can 'update' your copy.
The latest version of these scripts will be available via anonymous ftp on ftp.ifh.de:pub/hepix and via AFS on /afs/cern.ch/user/t/taddei/public[5]

The packages have the following names.

[5] This will change in the future

HEP_startup_files.tar.Z	for the login and interactive shell scripts
X_environment.tar.Z	for the X environment
keybindings_emacs19.tar.gz	for the emacs key bindings

In case of problems or wishes for modifications send a mail to koehler@ifh.de or taddei@dxcern.cern.ch. In order to unpack those files, `uncompress` enables you to decompress `.Z` files, `gunzip` can decompress `gz` files and mainly `tar xvf` enables you to de"tar" `.tar` files.

8 Conclusion

The aim of this paper was to propose an environment which is standardized but which provides nevertheless great flexibility for system administrator and for users.

The login scripts were tested on a large amount of different systems and in the different situations where the shell environment comes into the play. It lead to a more homogeneous system through the definition of a limited number of environment variables and aliases.

The main point for the definition of the proposed X environment was to have a small number of files the users and the system administrators have to manage. The proposed way to start an X session is the X Display Manager. With these X scripts the user has the flexibility to customize his X environment with standardized files.

Appendix A - Keyboard names

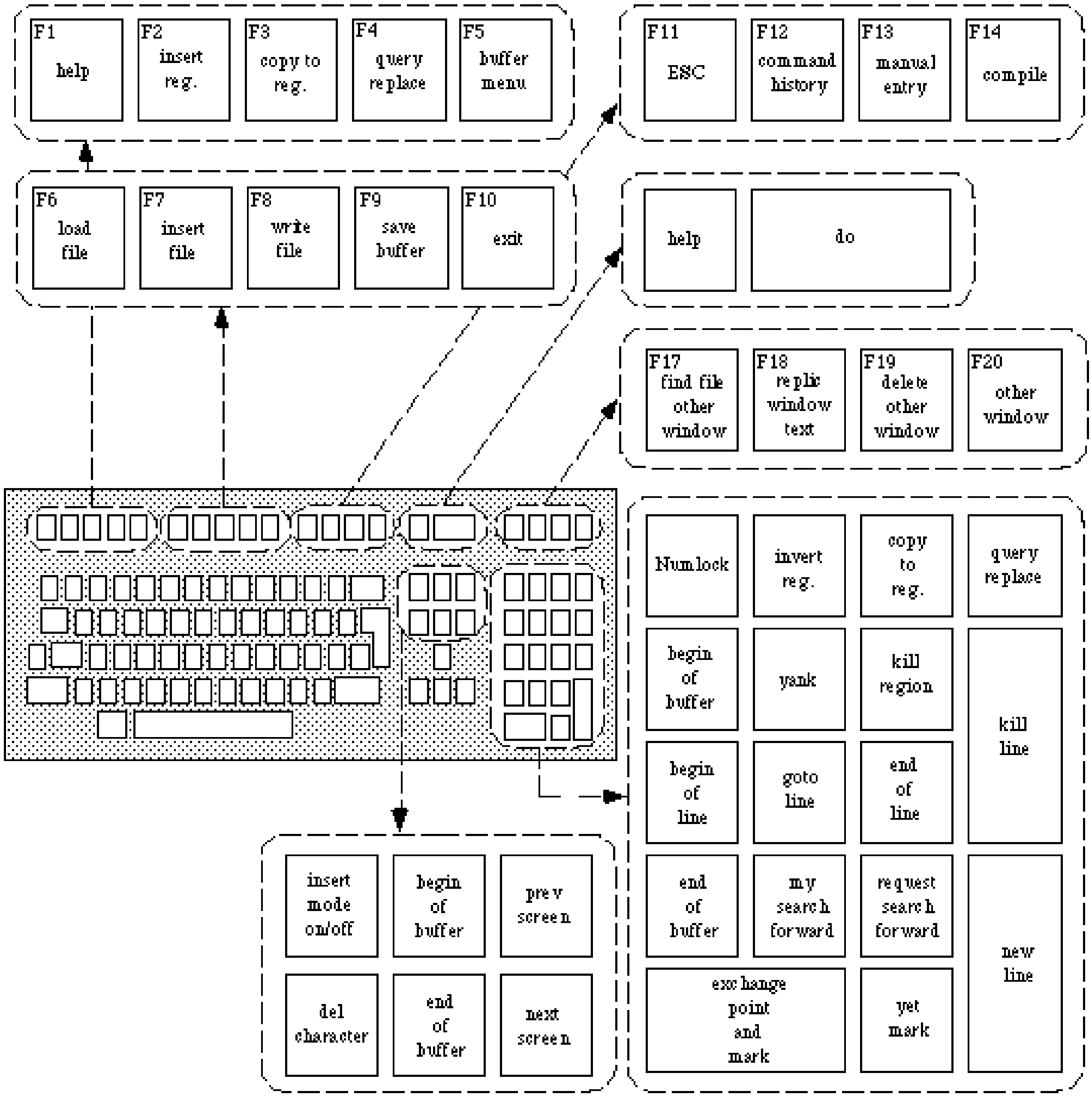
```
Keyboardname - Keyboarddescription
--------------------------------------------------------
Sun3         - Sun type3 (MIT layout)
Sun4         - Sun type4 (MIT layout)
Sun4ow       - Sun type4 (OpenWindows layout)
Sun5         - Sun type5 (MIT layout)
Sun5ow       - Sun type5 (OpenWindows layout)
Sun5swfr     - Sun type5 (Swiss-French layout)
Sun5pc       - Sun type5/PC (MIT layout)
Sun5pc2      - Sun type5/PC (MIT layout + patch)
Sun5pcow     - Sun type5/PC (OpenWindows layout)
Sun5pcowde   - Sun type5/PC (German OpenWindows layout)
NCD97        - Network Computing Devices N97
NCD101       - Network Computing Devices N101
NCD102       - Network Computing Devices N102 (US layout)
NCD102F      - Network Computing Devices N102 (French layout)
NCD102N      - Network Computing Devices N102 (Norwegian layout)
NCD102SF     - Network Computing Devices N102 (Swedish/Finnish layout)
NCD108       - Network Computing Devices N108
NCD220       - Network Computing Devices vt220
NCDPC        - Network Computing Devices PC-Xview
SGIR4        - Silicon Graphics (X11R4 layout)
SGI          - Silicon Graphics (X11R5 layout)
NWS          - Sony Net Work Station 1250
Mac          - Apple Extended II
GA5k         - Concurrent Computer Corp GA5000
LK201        - Digital Equipment Corporation LK201
LK401        - Digital Equipment Corporation LK401
LK421        - Digital Equipment Corporation LK421
RS6k         - IBM RS/6000 (AIXWindows R4 layout)
RS6kDE       - IBM RS/6000 (AIXWindows German R4 layout)
RS6k-MIT     - IBM RS/6000 (MIT X11R5 layout)
IBM130       - IBM XStation 130
HP-ITF       - Hewlett-Packard 300/400/700 Series
HP700RX      - Hewlett Packard 700/RX X Terminal
HPPC         - PC 101 key keyboard #1 (Hewlett-Packard PC US layout)
SCO110       - PC 101 key keyboard #1 (Santa Cruz Operation US layout)
DELL         - PC 101 key keyboard #1 (DELL PC US layout)
HCLPC        - PC 101 key keyboard #1 (Hummingbird eXceed US layout)
```

```
TT          - Atari TT
TEK101      - Tektronix XP20 101-Key (US X11R5 layout)
TEK101-4    - Tektronix XP20 101-Key (US X11R4 layout)
TEK200US    - Tektronix VT200 (US Ultrix layout)
TEK200DA    - Tektronix VT200 (Danish Ultrix layout)
TEK200DE    - Tektronix VT200 (German Ultrix layout)
TEK200FI    - Tektronix VT200 (Finnish Ultrix layout)
TEK200FR    - Tektronix VT200 (French Ultrix layout)
TEK200IT    - Tektronix VT200 (Italian Ultrix layout)
TEK200KA    - Tektronix VT200 (Katakana Ultrix layout)
TEK200NO    - Tektronix VT200 (Norwegian Ultrix layout)
TEK200SP    - Tektronix VT200 (Spanish Ultrix layout)
TEK200SW    - Tektronix VT200 (Swedish Ultrix layout)
TEK200SWDE  - Tektronix VT200 (Swiss-German Ultrix layout)
TEK200UK    - Tektronix VT200 (UK Ultrix layout)
TEK200VMS   - Tektronix VT200 (US VMS layout)
TEK200ESC   - Tektronix VT200 (US ESC layout)
TEK200F11   - Tektronix VT200 (US F11 layout)
Labtam      - Labtam X Terminal MT200
NeXT        - NeXTstation non-ADB
MAG3k       - MIPS Magnum 3000
MAG-AT      - MIPS Magnum AT (US layout)
RS2030      - MIPS RS-2030
Explorer    - Texas Instruments Explorer
```

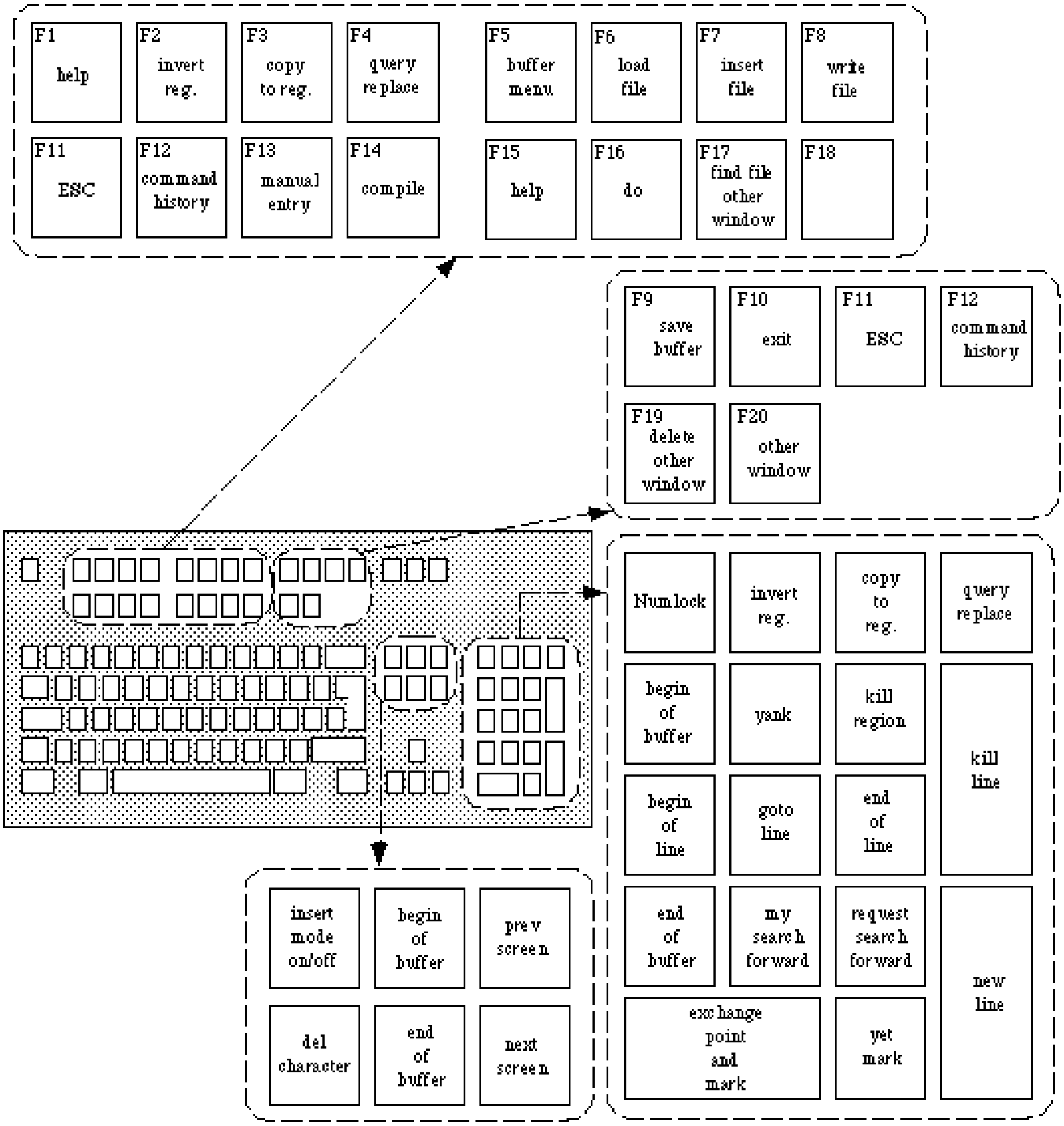
Appendix B - Motif window manager configuration

```
The motif window manager should be configured as follows:
---------------------------------------------------------
 Meta key is              Alt     on standard keyboards
                          Repeat  on old apollo keyboards
                          Compose on vt220 keyboards
                                  On new NCD VT200 keyboards
                                  Alt and Compose do the same
                                  functions

 Key/Button                       Action

 Button1 in root window:          GlobalMenu
 Button2 in root window:          GroupMenu
 Button3 in root window:          UserMenu
 <Alt>Button1:                    Select next window
 Shift<Alt>Button1:               Select previous window
 <Alt>Button2:                    move window or icon
 <Alt>Button3:                    Toggle back and front windows
                                  (Function:raise or lower)
 Double click on window frame:    Minimize window or normalize icon
 <Alt>Space,<Alt>Select:          Show window or icon menu
 Shift<Alt>Space,Shift<Alt>Select:Show GlobalMenu
 <Alt>Tab:                        Select next window or icon
 <Shift>Pop or <Alt>PrevSreen:    Select previous window
 <Alt><Shift>Tab:                 Select previous window or icon
```

Appendix C - Emacs keyboard mapping for VT xxx style keyboards



Appendix D - Emacs keyboard mapping for IBM PC style keyboards



References

[1] Lionel Cons. *CERN Security Handbook*. CERN, Geneva, version 1.0 edition, 2 December 1993. Practical computer security for CERN users.

[2] Lionel Cons. *X Terminal Administrator's Guide*. CERN, Geneva, version 1.0 edition, 6 April 1993. How to manage X terminals at CERN.

[3] Linda Mui and Eric Pearce. *X Window System Administrator's Guide*. O'Reilly & Associates, Inc., USA, 1992.