Contents

1 Introduction.............................................. 2

2 Overview of computing facilities....................... 3
  2.1 Machines............................................. 4
  2.2 Networks............................................. 4
  2.3 Principal software.................................... 5

3 Registration............................................. 5

4 Accounting............................................... 6

5 Access to ECMWF.......................................... 7
  5.1 Security tokens........................................ 7
  5.2 SSH connection........................................ 7
  5.3 Access to HPCF........................................ 8
  5.4 Access to ECMWF website............................. 9

6 Shells.................................................... 9
  6.1 Ecgate server......................................... 9
  6.2 HPCF.................................................. 9

7 File systems............................................. 11
  7.1 Server file systems.................................. 13
  7.2 HPCF file systems................................... 13
  7.3 NFS mounted file systems............................ 13
  7.4 ECFS.................................................. 14

8 File transfer........................................... 15
  8.1 Between machines within ECMWF...................... 15
  8.2 From ECMWF to external sites...................... 15
  8.3 From external sites to ECMWF....................... 18
  8.4 Between ECMWF file systems and tape.............. 18

9 Batch jobs.............................................. 19
  9.1 Job submission....................................... 19
  9.2 Multi-processor jobs................................ 22
  9.3 Examples............................................ 23
  9.4 Submission of time-critical jobs................... 23
  9.5 SMS.................................................. 23

10 Fortran................................................ 23
  10.1 Compiler invocation.................................. 24
  10.2 Compiler options.................................... 25
    10.2.1 Standard options............................... 25
    10.2.2 Options that control Fortran source format .. 26
    10.2.3 Listings options............................... 26
    10.2.4 Compatibility options.......................... 26
    10.2.5 Optimization options............................ 27
1 Introduction

This document describes those ECMWF computing facilities relevant to external users. First you will find an overview of the different machines and their functions at ECMWF, plus a description of the networks linking these machines. Then there is a brief summary of the principal software available. Finally, we focus on various topics of particular interest when using ECMWF's systems.

More detailed documentation and various services are available from the ECMWF website

www.ecmwf.int

... card. Most parts of this website can be accessed by users recognized by their Internet domain address (e.g. users from a Member State or Co-operating State meteorological service). For specific web services (e.g. prepIFS, full MARS access) or for users not recognized by their domain address (e.g. users from universities), authentication via a Web Certificate or by user-id and passcode is required. See Section 5.4 for how to access the ECMWF website.

An on-line version of this document is available at

www.ecmwf.int/services/computing/help/new_user/intro_ex/
2 Overview of computing facilities

This section gives an overview of the main hardware installed in ECMWF’s computer hall (see Figure 1) and summarizes the principal software relevant to external users. Additional information can be found at

www.ecmwf.int/services/computing/overview/

Figure 1: Network and computer systems configuration (Feb 2010).
2.1 Machines

IBM Cluster 1600

The High Performance Computing Facility (HPCF) consists of two independent clusters, called c1a and c1b, based on pSeries p6-575 servers interconnected by a low latency high speed infiniband network. Each cluster comprises 262 (+8) application nodes with 64 (256) GB memory, 12 network and I/O routing nodes and a further 4 nodes used as spare and service nodes. As the Power6 processors are capable of simultaneous multi-threading (SMT), which allows to run 2 threads concurrently, the nodes appear to have 64 (virtual) CPUs rather than 32 physical processors.

The whole system has about 1.2 petabyte of SAS (Serial Attached SCSI) disks, connected to two separate storage I/O clusters which serve their data to both compute clusters over an infiniband network, to which the 12 network nodes in each compute cluster are connected.

The operating system is AIX, IBM’s Unix variant. For more information refer to the HPCF pages at www.ecmwf.int/services/computing/hpcf/.

DHS

The Data Handling System comprises MARS, the meteorological archive, and ECFS, the user archive. The DHS is based on HPSS (High Performance Storage System), where the bulk of data is kept on tapes stored in automated tape libraries (ATLs).

During the course of this year the 5 STK silos will be replaced by 3 Sun StorageTek SL8500 ATL, which will have licensed storage capacity to 24 PB. Other main components of the DHS are various servers (IBM p650, p660 and p570) and several TB of disk cache.

Servers

Various general purpose and high availability servers support our users. While the IBM p5 575 cluster ecgate, running AIX, provides a service exclusively to registered external users, internal users have access to a Linux Cluster.

External users, via ecgate, and internal users, via a Linux based PC, can

- work in interactive mode to edit files, prepare jobs, run graphics applications, etc. and
- submit batch jobs to the servers and the HPCF

For more information on the Member State server ecgate refer to www.ecmwf.int/services/computing/ecgate/

The Hewlett Packard RX4640 servers are used for operational Data Acquisition, Preprocessing and Dissemination. A second high availability cluster, HP L-Class, is used as NFS server.

2.2 Networks

The Local Area Network (LAN) connects all ECMWF’s computing systems. The LAN is entirely based on Ethernet technology and segmented into a High Performance Network (HPN) and a General Purpose Network (GPN).

HPN

The HPN is used for the transfer of large amounts of operational data. It connects the HPCF and the DHS via multi-Gigabit-Ethernet.

GPN

The GPN is used for all other transfers, providing connectivity to:
2.3 Principal software

- The HPCF, the DHS and additional large servers (Gigabit Ethernet)
- The user desktops and laptops (Ethernet 10/100/1000).
- The firewalls (for the Wide Area Network and the Demilitarized Zone; Ethernet 10/100/1000)

C Compilers are available on the ecgate servers and the HPCF; documentation on the C compilers is available via the man pages (`man cc`).

ECFS The ECMWF File Storage system provides storage for your data/files. It is directly accessible from all Unix systems. See Section 7.4 for more details.

FORTRAN Compilers are available on the ecgate server and the HPCF; all systems accept Fortran 90 and Fortran 77. See Section 10 for more details.

LIBRARIES Various software libraries, which contain commonly used routines, are supported on the IBM servers and the HPCF, e.g. EMOSLIB, ECLIB, NAG. For more details see Section 11.

MAGICS ECMWF’s library of graphics routines (not available on the HPCF) (see Section 12).

MARS ECMWF’s Meteorological Archival and Retrieval System; MARS is accessible from all the major systems at ECMWF, as well as from the ECMWF Web Server (`www.ecmwf.int/services/archive/`). It contains observations, analyses and forecasts, both from the ECMWF operational forecast system and from various research experiments. In addition, special data sets are available, e.g. ECMWF Re-Analysis data.

For an introduction to MARS please refer to the MARS User Guide, which is included in the New User Pack and also available at `www.ecmwf.int/services/computing/docs/archives/mars/`

METVIEW ECMWF’s interactive visualisation package that enables operational and research meteorologists to access, manipulate and visualise meteorological data on Unix workstations (see Section 12);

UNIX Mostly based on the standard version V release 4; it includes all the normal Unix utilities such as vi, awk, etc.; some common public domain utilities are also available e.g. perl. ECMWF’s preferred Unix shell is the Korn (POSIX) shell (See Section 6).

3 Registration

Every user of the ECMWF computer system is given an individual 3 character user identifier (uid).

Security requirements demand that a given identifier must not be shared by two or more users. ECMWF reserves the right to disable a user identifier if sharing has occurred.

The first 2 characters of the user identifier are common to a given group or section (e.g. all User Support identifiers begin with `us`). The third character is assigned by your Computing Representative or Special Project Principal Investigator, as appropriate.

At registration each uid is assigned to one or more groups. By default, all users in a group can read each others’ files, whereas files belonging to other groups are not automatically accessible (`umask=027`). All users in a Member State or Co-operating State are assigned by default to the same group, except for Special...
Project users, who are assigned to a group unique to their Special Project. If it is required that a set of users is able to share the files of a group which is not the default group, User Support should be contacted, preferably at the time of registration. Groups of users are assigned to a project or account. Note that all users under a given account will usually be assigned to the same groups. You can see the list of your groups by using the command

```
  groups [ uid ]
```

The first group listed is the default group. Any file you create is automatically assigned to that group first. Use the command

```
  chgrp newgroup filename
```

to change the assignment of the existing file(s) `filename` to `newgroup`.

## 4 Accounting

Each project has a unique account name. Every job run on the HPCF systems incurs a charge on its individual account. The charge is based on a System Billing Unit (SBU) formula. SBU accounting is used to control system usage, the main points being:

- Every 24 hours SBU usage on every project is calculated and added to the total usage, cumulative from the beginning of the accounting year.
- The total usage is compared to an annual allocation for the project. If the usage exceeds the allocation, then no further use will be allowed under that project, unless additional resources are transferred.
- Unused resources can be moved from project to project by contacting User Support (ECMWF staff) or your Computing Representative.

Reports listing HPCF usage against each project are generated at regular intervals (daily, weekly, 4-weekly, annually), the 4-weekly ones being widely distributed. The `acct_status` command can be used to show the SBU usage for your project(s).

For a full description of the charging and control system see

```
  www.ecmwf.int/services/computing/hpcf/accounting.html
```

To see the list of accounts on which your `uid` is registered, use the command

```
  account [ -l uid ]
```

Where a `uid` is registered on more than one account, the first account on this list is its default account. If you do not specify a particular account on a HPCF job, then all usage is automatically debited to the default account. To charge usage to an alternative account, you must specify the LoadLeveler keyword

```
  #@ account_no = alt_account
```

where `alt_account` is the alternative account. If you specify an invalid account, your job will be rejected at submission time.
5 Access to ECMWF

The actual method of access to the ECMWF server ecgate may vary from country to country. Contact your Computing Representative for information on the methods available to you. In any case, you need a security token, which will either be in your New User Pack or will be provided by your Computing Representative.

The various methods of access to ECMWF are described below (note that the actual IP addresses and host names used will be different from the ones shown in the examples below). For information on file transfer (ftp) please see Section 8.

If you have any problems with access to the ECMWF systems, please contact ECMWF’s 24 hour Call Desk (cdk@ecmwf.int).

Note: System sessions are usually scheduled on Wednesday mornings. During these system sessions you can expect disruptions or unavailability of some services. Please keep yourself informed about planned system sessions and other events by using the cos_info command on ecgate (type man cos_info) or by following the link Cosinfo from www.ecmwf.int/services/computing/

5.1 Security tokens

Actividentity security tokens are needed for all logins to ECMWF machines from external sites. The security token generates passcodes, also referred to as one-time passwords (OTP), which are then used in place of passwords. Security tokens are issued and administered by your Computing Representative or User Support.

More information can be found in the “Actividentity token leaflet”

www.ecmwf.int/services/computing/help/access/actividentity_ref.pdf

5.2 SSH connection

The ECaccess gateway provides interactive access to the ECMWF computing facilities via the Internet for registered users. The gateway is one component of ECaccess, a framework for batch and interactive access to ECMWF services for Member State and other external users.

Gateways include a model for the management of “plugin” services. A plugin is a piece of code that handles requests/responses flowing through the gateway.

The following applies to all Secure Shell (SSH) access.

- For connection via Internet and RMDCN symmetrical gateways are available:
  - ecaccess.ecmwf.int for connections via Internet
  - msaccess.ecmwf.int for connections via RMDCN

In the following examples ECaccess and MSaccess can be used interchangeably. Note that telnet connections to ecaccess.ecmwf.int and msaccess.ecmwf.int are not allowed. Please ask your Computing Representative or System Administrator how you are linked to ECMWF.

- Local gateways may be installed at your site (e.g. ecaccess.meteo.ms); if so, they should be used in preference to the ECMWF gateways. To local gateways you can also connect with telnet. Please ask your Computing Representative or System Administrator which options are available to you.
• For each login your ECMWF user-identifier and token passcode are necessary. Please remember that after you have entered 10 wrong PINs the token will be locked.

• After login you will get a choice of systems you can directly connect to.

For comprehensive information on the ECaccess concept and all available access facilities please refer to the documentation at

www.ecmwf.int/services/ecaccess/

The ECaccess SSH plugin allows registered users to log into their ECMWF account, by using the ssh command to connect to a gateway. With the -X option you can enable X11 forwarding. Then you will be asked for a token passcode. At the final prompt you can choose the system you want to connect to.

Please note that the actual host names used may differ from the ones shown in the following example session:

```bash
%ssh -l uid -X ecaccess.ecmwf.int
Authorized access only.

***********************************************
... banner ...
***********************************************
Password authentication
uid’s password

***********************************************
... help text ...
***********************************************
Select hostname (ecgate, c1a) [ecgate]:
```

An alternative service using the NX technology allows users to run remote X Window sessions even across slow or low-bandwidth network connections, making it possible to start sessions from clients running on Windows, Linux, Mac OS X and Solaris platforms.

The easiest way to use this service is via a web browser but it can also be used via a standalone NX client. For more information please refer to the documentation at

www.ecmwf.int/services/ecaccess/guide/NX_service.html

5.3 Access to HPCF

The HPCF should be used for computing intensive work in batch mode. Computing intensive interactive work is not recommended. However, if you do need to access these machines interactively, e.g. to submit a LoadLeveler script or to check the queue, you can directly login to c1a. Follow the procedure given in Section 5.2 for the SSH connection and when asked for a hostname specify c1a. Alternatively, you can first login to ecgate and from there you can login to cluster c1a using
5.4 Access to ECMWF website

A growing number of forecast products, documents, services, etc. are accessible on the ECMWF website, www.ecmwf.int

Most areas of this website can be accessed by users recognized by their Internet domain address (e.g. users from a Member State or Co-operating State meteorological service). For specific web services, e.g. prepIFS, full MARS access, and for users not recognized by their domain address, e.g. users from universities, login to the website using a Web Certificate is recommended. Certificates can be applied for at the ECMWF Certificate Authority, http://w3cert.ecmwf.int/ (Security tokens need to be unlocked).

To check how you are recognized by the ECMWF Web Server and to make sure you are logged in according to your privileges click on the Login link at the very top of any page. From there you can also choose to login using your ECMWF user-ID and passcode. This option is particularly recommended for occasional access or in case of technical difficulties with Web certificates. For a more information please refer to www.ecmwf.int/services/computing/help/access/web-access.html

6 Shells

When you login to a Unix machine or submit a batch job, your commands are interpreted by a login shell. Each time you login or open a shell, several configuration files are executed. These files define variables, set functions, etc., depending on the actual shell.

During the initial registration the login shell is set by the system administrator and some of the default configuration files are copied into your $HOME directory. Some configuration files are compulsory while others can be customized. In particular the .user_* files are intended for your own settings. Please do not remove calls or links to shared configuration files, as it can then no longer be guaranteed that your environment would be correct after system changes. Whenever you make changes to your configuration files, do not forget to execute them.

6.1 Ecgate server

The only supported command interpreters are the Korn shell (ksh) and the C shell (csh). For full functionality the Korn shell is recommended as the login shell. Other shells (Bourne, tcsh) may be available but are not fully supported.

Tables 1 and 2 list the configuration files for csh and ksh. The .user_* are only executed if present. On ecgate you can use the command setup_user to get an up-to-date copy of the configuration files.

6.2 HPCF

The login shell on the HPCF is forced to be the Korn shell. Common configuration files are implemented as symbolic links from the user’s home directory. Personal customization should be done in .user_* files.
Introduction to computing facilities

<table>
<thead>
<tr>
<th>file name</th>
<th>execution</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/profile</td>
<td>each login and job execution</td>
<td>unchanged system-wide environment settings</td>
</tr>
<tr>
<td>$HOME/.profile</td>
<td>each login and job execution</td>
<td>local environment settings, link to a shared file</td>
</tr>
<tr>
<td>$HOME/.user_profile</td>
<td>called by .profile</td>
<td>suitable for your own settings (e.g. PATH)</td>
</tr>
<tr>
<td>$HOME/.kshrc</td>
<td>each ksh startup</td>
<td>shared setups for ksh users, link to a shared file</td>
</tr>
<tr>
<td>$HOME/.user_kshrc</td>
<td>called by .kshrc</td>
<td>suitable for your own settings (aliases, variables, functions)</td>
</tr>
<tr>
<td>$HOME/.user_epilog</td>
<td>each logout and end of job</td>
<td>your own logout procedure</td>
</tr>
<tr>
<td>~ecmwf/share/.epilog</td>
<td>each logout and end of job</td>
<td>removal of temporary directories</td>
</tr>
</tbody>
</table>

Table 1: Korn shell configuration files on ecgate in the order of execution.

<table>
<thead>
<tr>
<th>file name</th>
<th>execution</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/cshrc</td>
<td>each login</td>
<td>unchanged system-wide csh initialization</td>
</tr>
<tr>
<td>$HOME/.cshrc</td>
<td>each csh startup</td>
<td>suitable for your own settings</td>
</tr>
<tr>
<td>$HOME/.login</td>
<td>each login</td>
<td>own settings (changing the shell)</td>
</tr>
<tr>
<td>$HOME/.logout</td>
<td>each logout</td>
<td>removal of temporary directories, link to a shared file</td>
</tr>
<tr>
<td>$HOME/.user_logout</td>
<td>called by .logout</td>
<td>own logout procedure, link to a shared file</td>
</tr>
</tbody>
</table>

Table 2: C shell configuration files on ecgate in the order of execution.

which are not provided by default. See Table 3 for the configuration files actually used.

<table>
<thead>
<tr>
<th>file name</th>
<th>execution</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HOME/.profile</td>
<td>each login and job execution</td>
<td>common environment settings</td>
</tr>
<tr>
<td>$HOME/.user_profile</td>
<td>called by .profile</td>
<td>suitable for your own settings</td>
</tr>
<tr>
<td>$HOME/.kshrc</td>
<td>each ksh startup</td>
<td>common setups for ksh users</td>
</tr>
<tr>
<td>$HOME/.user_kshrc</td>
<td>called by .kshrc</td>
<td>suitable for your own settings (aliases, variables, functions)</td>
</tr>
<tr>
<td>/usr/local/share/.epilog</td>
<td>at logout and end of job</td>
<td>accounting, removal of temporary directories</td>
</tr>
</tbody>
</table>

Table 3: Korn shell configuration files on the HPCF in the order of execution.
7 File systems

Under Unix the disk storage area is divided into file systems, each of which can be considered a separate storage area for files. File systems are created by the system administrators.

Within a file system files are stored in directories. At the top of each file system is a root directory, under which there are various system or user subdirectories. Directories can contain subdirectories and/or files, thus each Unix file system is in the form of a hierarchical tree structure.

In addition to the common Unix file manipulating commands, some locally developed commands are available.

At ECMWF the user storage area is divided into the following file system types (see also Figure 2):

- **Permanent**: The only permanent file system available to users to store their files in is the home file system ($HOME). Within your quota (see below) you can store files for as long as you wish. This area is backed up daily and, in the event of a disk failure or an unwanted file delete, it can be fully restored from the last available backup.

- **Temporary**: Files are kept as long as possible but are liable to be deleted at any time without warning when free space is low (see select/delete below). No backups are taken. Example: the scratch file system ($SCRATCH).

- **Automatically deleted**: Files are automatically removed at the end of your job or interactive session. These file systems are meant as work areas for utilities and packages or for your own temporary data. No backups are taken. Example: $SCRATCHDIR on ecgate.

- **ECFS (archive)**: The ECMWF File Storage system (see Section 7.4) provides storage for your data/files from both the server and HPCF systems. ECFS is intended as a long term archive and as a place for storing excess data/files which would otherwise cause your home file system quota to be exceeded.

The following items are important on all machines:

- **Quotas** restrict the disk space allocated to the user. Quotas are currently applied to $HOME on all platforms and to the $SCRATCH file systems on the server. There is a soft quota limit and a hard limit. If you exceed either of the limits, you will get an e-mail warning. From the time the system detects your overuse, you will usually have about a week to reduce your usage to below the soft limit. If you exceed the hard limit, the system will prevent you from creating any more files. To find out your current quota and space used (in kbyte blocks), use the command:

  ```
  ecquota # on ecgate
  quota # on HPCF
  ```

- **Select/delete** is a mechanism applied on all temporary file systems (not on $HOME) to ensure that a certain percentage of free disk space is available. On ecgate, once the disk usage exceeds a particular threshold, the oldest files are deleted without warning until there is 50% space free, while files newer than 3 days old are usually kept. To avoid unnecessary deletions by this mechanism, you should clear your space as soon as possible, by deleting your files when they are no longer required or by moving them to ECFS.

- **Backups** are made automatically only for $HOME directories. The main reason for backups is to protect against machine failure. Although it is possible to restore files from recent backups, if accidentally deleted by the user, it is not guaranteed.

February 12, 2010
Introduction to computing facilities

- Machines – accessible to users
- File systems – suitable for permanent files (‘small’ quotas)
- File systems – suitable for temporary data
- File systems – directories automatically deleted at end of job
- Local file systems
- Cross mounted file systems

Figure 2: Structure of ECMWF file systems.
Note:

- The $HOME file systems on the server and the HPCF systems are all different file systems. It is possible, however, through NFS, to have access to the other file systems from a given machine (see Section 7.3).

- You should copy all critical files from file systems other than $HOME to ECFS without delay.

### 7.1 Server file systems

On the server ecgate the permanent $HOME file system and the temporary $SCRATCH file system are available. $SCRATCH is a General Parallel File System (GPFS), suitable for holding the bulk of user files (e.g. working data). These files will be retained for as long as there is sufficient space. However, whenever the free space falls below a preset level, the select/delete mechanism will be activated. The $SCRATCHDIR directory is a subdirectory of $SCRATCH and is automatically deleted at the end of each batch job or interactive session (see Table 4).

<table>
<thead>
<tr>
<th>File System</th>
<th>Suitable for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HOME</td>
<td>permanent files, e.g. .profile, utilities, sources, libraries</td>
</tr>
<tr>
<td>$SCRATCH</td>
<td>all temporary files</td>
</tr>
<tr>
<td>$SCRATCHDIR</td>
<td>data to be automatically deleted at end of job</td>
</tr>
</tbody>
</table>

*Table 4: Unix file systems on the ecgate server. Note that $SCRATCHDIR is part of $SCRATCH.*

### 7.2 HPCF file systems

On the HPCF (see Table 5) all user file systems are of the type GPFS (General Parallel File System). GPFS is a high performance, high-availability parallel file system which allows data to be spread across multiple nodes. It can be used by all parallel and serial applications. To achieve the best I/O performance files on $TEMP should be accessed sequentially with block-sizes of multiples of 4 MB.

<table>
<thead>
<tr>
<th>File System</th>
<th>Suitable for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HOME</td>
<td>permanent files, e.g. .profile, utilities, sources, libraries</td>
</tr>
<tr>
<td>$TEMP</td>
<td>all temporary files</td>
</tr>
<tr>
<td>$TMPDIR</td>
<td>data to be automatically deleted at end of job</td>
</tr>
</tbody>
</table>

*Table 5: File systems on HPCF. All file systems are of type GPFS. Note that $TMPDIR is part of $TEMP.*

### 7.3 NFS mounted file systems

It is possible, via a software known as NFS (Network File System), to have the same file system available from many different machines. The files are physically held on one system, say on ecgate, and then mounted on another e.g. c1a. When a command to read or write a file is issued on c1a, blocks of that file are transferred transparently between the server and c1a, as required. You are not aware that the file you are working with is actually on another machine.
File systems can be statically mounted or automounted. The automounter is a daemon that automatically and transparently mounts any NFS volume as needed, and unmounts it after it has not been used for some time. Which file system a machine is allowed to access is defined in NIS (Network Information System) maps, which can be displayed using the `ypcat` command, e.g. `ypcat -k auto.master`. The `mount` command will tell you which file systems are currently mounted.

Although this is a very convenient way of accessing the same file from two different machines, as all the standard Unix file commands are available, without the need to have a local copy of the file, it is not suitable for transferring large amounts of data.

Also beware: file locking is not done automatically, so strange results can occur, if the same file is, for instance, written to on one machine and read simultaneously from another.

Using NFS the main file systems are available on the HPCF systems and all servers. To avoid problems when working with these common file systems and especially to prevent confusion as to which home file system you are using, always use their complete path name.

The NFS mounted file systems are listed in Table 6, where `gid` is your Unix primary group identifier, e.g. ch for Swiss users, and `uid` is your user identifier.

<table>
<thead>
<tr>
<th>name on server</th>
<th>remote name</th>
</tr>
</thead>
<tbody>
<tr>
<td>/c1a$HOME</td>
<td>/home/ms/gid/uid on c1a</td>
</tr>
<tr>
<td>/hpcf$HOME</td>
<td>/home/ms/gid/uid on hpcf</td>
</tr>
</tbody>
</table>

Table 6: NFS mounted file systems on the server.

<table>
<thead>
<tr>
<th>name on HPCF</th>
<th>remote name (on ecgate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ws/home/ms/gid/uid (or $WSHOME)</td>
<td>/home/ms/gid/uid</td>
</tr>
<tr>
<td>/ws/scratch/ms/gid/uid (or /ws$SCRATCH(^1))</td>
<td>/scratch/ms/gid/uid</td>
</tr>
</tbody>
</table>

\(^1\) If \$SCRATCH is defined

Table 7: NFS mounted file systems on the HPCF interactive node (only).

7.4 ECFS

The ECMWF File Storage system (ECFS) is available as part of the Data Handling System (DHS, see Figure 2). It provides storage for your data/files from both the HPCF and workstation/server systems. ECFS is intended both as a long term archive and as a place for storing huge amounts of data which would cause your home file system quota to be exceeded. It is not meant as an alternative \$SCRATCH. Please also note that ECFS usage is accounted. To check on your ECFS usage use the `ecfs_status` command.

ECFS is a Unix based file system accessible by Unix-like commands, which means that ECFS commands add an e prefix to the Unix command, e.g. `ecp` to copy a file. The set of basic commands available is: els, erm, ermdir, ecd, epwd, echmod, ecp and emv. With the `emove` command files can be moved within one ECFS system. The command `ecfsdir` offers an easy way to save or retrieve a complete UNIX directory as one ECFS file. ECFS commands can be used interactively and in batch scripts, in both the Korn (POSIX) and C shell environments.

Please note that *any ECFS files removed (deleted) by a user cannot be recovered.*

For further information see the man pages or the ECFS pages at

www.ecmwf.int/services/computing/docs/archives/ecfs/
8 File transfer

For a file transfer between external sites and ECMWF, ECaccess, a portal to access ECMWF’s computing and archiving facilities, is the recommended tool which should cater for most of your needs. For transfers initiated from ECMWF using the `ectrans` command, the functionality available to you will depend very much on whether or not you have a local ECaccess/MSaccess gateway installed. Please contact your Computing Representative to get more information on the ECaccess components installed at your site.

For more information on the ECaccess concept and functionality and also to download the software, please refer to the documentation at

```
www.ecmwf.int/services/ecaccess/
```

The following subsections describe file transfers within ECMWF and between ECMWF and external sites, emphasizing alternative methods for when a local ECaccess gateway is not available.

### 8.1 Between machines within ECMWF

For data transfers between server and HPCF the `ecrcp` command, which is a locally developed adaptation and optimization of the RSYNC software, is recommended.

Using a compatible syntax it offers the same functionality as the standard `rcp` command but has many more options and is more robust and thus better suited for batch applications. Since RSYNC transfers just the differences between two sets of files across the network, it is most efficient when the destination file already exists. The following example issued on ecgate will copy files from your c1a `$HOME` directory to your `$SCRATCH` directory on ecgate without overwriting newer files:

```
ecrcp -u c1a:filename $SCRATCH/
```

It is always advisable to initiate such transfers from the server rather than from the HPCF.

### 8.2 From ECMWF to external sites

Various methods are available to transfer files from ECMWF to remote sites. Which one is best suited for your requirements depends to a great extend on the configuration of the remote system. If you are not sure about that, please check with your Computing Representative or system administrator.

In general you have the following options:

- **ectrans**

  The `ectrans` command, part of the ECaccess framework, allows files to be transferred securely and unattended as it does not require a password to be specified for the remote host: the ECaccess gateway performs the security checking. Ideally, an ECaccess gateway should be installed at the remote site to which you want to transfer to. But even if this is not the case, you can use ectrans via the ECMWF gateway (ecaccess.ecmwf.int), if your destination site is accessible from the ECMWF gateway via (s)ftp.

  Before you can make use of ectrans, you need to declare a remote Member State user (`msuser`) for the storage/retrieval of the remote file. This has to be done through the ECaccess Web interface of the gateway you want to transfer to (or at [http://ecaccess.ecmwf.int](http://ecaccess.ecmwf.int) if no gateway is installed at...
the remote site). For every msuser declaration, the hostname and the login username and password are requested and then stored on the gateway in encrypted form.

After these preliminaries you should be able to use ectrans as described in its help page:

```
ecgate$ ectrans -help
usage: ectrans [-gateway name] -remote msuser@[destination] \ 
       [-get|-put] -source [ec:|ectmp:]filename [args ...] (*) 
       ectrans -check requestID (*) 

-gateway {arg} - access gateway name (default (**): ecaccess.ecmwf.int)
-remote {arg} - access method (default (**): *none*)
-source {arg} - source file name
-target {arg} - target file name (default: same as -source)
-mailto {arg} - target email address (default: current user)
-lifetime {arg} - lifetime of the file in the spool (default: lw) (***) (***)
-delay {arg} - transmission delay (default: immediate transfer) (***) (***)
-at {arg} - transmission date (default: immediate transfer) (***) (***)
-format {arg} - define the date format as used with -at (default: yyyyMMddHHmmss)
-retryCnt {arg} - define the number of retries (default: async=144, sync=0)
-retryFrq {arg} - define the frequency of retries (default: async=10m, sync=1m) (***)
-priority {arg} - transmission priority 0-99 (default: 99) (***)
-put - interactive/synchronous transfer (no spool)
-get - interactive/synchronous pull (rather than push) file
-onsuccess - mail sent on successful transfer
-onfailure - mail sent when transfer has failed
-onretry - mail sent when transfer is retried
-keep - keep the request in the spool till expiration (***) (***)
-remove - always remove the request from the spool (***) (***)
-reject - if existing target file (default)
-append - if existing target file
-resume - if existing target file
-overwrite - if existing target file
-verbose - verbose mode on
-version - print version number
-help - this message

(*) If successful, a requestID is returned, which can be used in check requests. Exit code is 0 on success and >0 otherwise.
(**) The default values depend on the GATEWAY or REMOTE environment variables.
(*** Duration in weeks, days, hours, minutes or seconds (e.g. lw|2d).
(****) These options are only relevant when the spool is used. The spool is no used during interactive transfers (-get and -put options).
(***** By default, successful requests are removed from the spool and failed requests are kept in the spool till expiration.

Ectrans can be regarded as an extended ftp, which offers e.g. direct access to ECFS files, restart facility for failed transfers etc. For more details please refer to the ECAccess Web pages at

www.ecmwf.int/services/ecaccess/

or contact User Support.

sftp/scp

On ecgate and HPCF sftp and scp are fully supported as part of SSH to directly connect to remote sites via the Internet.
ftp

Only if connected via RMDCN to a meteorological service in a Member State or Co-operating State, FTP (File Transfer Protocol) can be used from ecgate without a proxy.

ftp proxy

To connect to *any external Internet site* with FTP, you always have to go through our ftp proxy. To send your user name and password automatically, you should add a line such as

```
    machine proxy login <username@ms_host> password <passwd>
```

to your `.netrc` file which has to be readable only by you. Alternatively, an interactive ftp connection from ecgate to an external host would look like this:

```
--> ftp proxy
Connected to proxy.ecmwf.int.
220- 136.156.64.10 PROXY-FTP server (DeleGate/9.9.5) ready.
220- @ @
220- ( - ) { DeleGate/9.9.5 {September 30, 2009} }
220- Copyright (c) 1994-2000 Yutaka Sato and ETL,AIST,MITI
220- Copyright (c) 2001-2009 National Institute of Advanced Industrial Science and Technology (AIST)
220- WWW: http://www.delegate.org/delegate/
220- You can connect to a SERVER by 'user' command:
220- ftp> user username@SERVER
220- or by 'cd' command (after logged in as an anonymous user):
220- ftp> cd //SERVER
220- Cache is enabled by default and can be disabled by 'cd .' (toggle)
220- This (proxy) service is maintained by 'wan@ecmwf.int'
220- extended FTP [MODE XDC][XDC/BASE64]
220 Name (proxy:usc): username@ms-host
331 Password required for username.
Password:
230-- PASS for username@ms-host.
220-Welcome to Pure-FTPd.
220-You are user number 2 of 50 allowed.
220-IPv6 connections are also welcome on this server.
220-You will be disconnected after 15 minutes of inactivity.
331 User username OK. Password required
230-User username has group access to: 500
230-OK. Current directory is /.
230 \(- )/ -- ( connected to 'ms-host' )
ftp>
ftp> get filename
...
ftp> quit
```

If you have added your username and password to your `.netrc` file, as described earlier, you can also integrate the ftp connection into a batch job:

```
... #@ queue
#
ftp proxy <<EOF
```

February 12, 2010
When transferring data, make sure you are using the right transfer type (binary/ASCII). Attempting to transfer a binary file as ASCII will result in a corrupted binary file.

The drawbacks of this method are that you can connect to only one external site and that you have to store/transfer your password. This is, in addition to other benefits, avoided if you use the command `ectrans` as described before.

### 8.3 From external sites to ECMWF

For a data transfer from an Internet site to ECMWF, invoke a connection to the ECaccess gateway by typing

```
ftp ecaccess.ecmwf.int
```

and providing your uid and security token passcode. To do such a transfer from a site connected to the RMDCN use

```
ftp msaccess.ecmwf.int
```

The ECaccess FTP plugin is an extended FTP server adding features to submit jobs to ecgate/HPCF or to exchange files with ECFS directories.

For more information on the ECaccess concept and on extended ftp commands, e.g. how to use ftp transfers in batch, please refer to the documentation at

[www.ecmwf.int/services/ecaccess/](http://www.ecmwf.int/services/ecaccess/)

Please note that sftp to ecaccess is not supported.

### 8.4 Between ECMWF file systems and tape

Insufficient bandwidth on RMDCN or the Internet may lead you to use other transfer media. Currently a service to write ECFS files on DVD (4.7 GB), 4mm DAT (2 GB) and DLT IV cartridges (10 - 70 GB) is provided. For this service, please send a request to the `tape-library@ecmwf.int` containing

- your postal address
- ECFS pathname
- list of files
- estimate of data volume
- type of media (DVD/DAT/DLT) with relevant information on density/compression

Please note that you must provide the Tape Library with the required media in advance. Remember to give read access to the tape librarian, i.e. use the command `echmod 644 filename` (owner has read and write, group and world have read permission).
9 Batch jobs

The processing of jobs in batch mode allows for better usage of the available limited resources compared to interactive mode, in particular for multi-processor jobs, and hence gives a better service to users. On ecgate and HPCF the IBM queuing/scheduling system LoadLeveler will allow you to submit your jobs, check the queue status etc.

On both systems you will need a normal script file with a special set of header lines (which are interpreted as comments, if the script is executed interactively) that define various parameters e.g. job name, time limit etc. It is common practice to submit batch jobs from ECMWF’s machines but it is also possible to submit jobs directly from a remote host using ECaccess.

Please note that all parallel jobs should be submitted to the HPCF.

9.1 Job submission

On ecgate and the HPCF the IBM batch job scheduling application LoadLeveler is used. Potentially, it can match your job requirements with the best available resources in a pool, which can be a collection of available machines. However, submission across systems is currently not possible, as each system runs its own LoadLeveler: you have to submit and control a jobscript on the system on which you want to run it.

Every LoadLeveler batch job must be defined by a command file in which various keywords are set. These keywords are identified by lines beginning with `@`. Although the same keywords can be used on both systems, the set of recommended keywords and the available job classes (queues) differ. See 3 and Figure 4 for example scripts suitable for ecgate and the HPCF, respectively.

Upon submission your command file is passed through a filter. If your job is ill-defined, it will be rejected before it even gets to LoadLeveler and additional information will be written to the terminal.

Apart from special classes available only to specific groups (such as for operational running), the main user classes are given in Table 8.

<table>
<thead>
<tr>
<th>System</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecgate</td>
<td>express</td>
<td>short jobs, e.g. compilations, access to real-time data</td>
</tr>
<tr>
<td></td>
<td>normal</td>
<td>default class, suitable for most batch work</td>
</tr>
<tr>
<td></td>
<td>large</td>
<td>jobs with large memory requirements</td>
</tr>
<tr>
<td></td>
<td>long</td>
<td>long and/or large jobs</td>
</tr>
<tr>
<td>c1a</td>
<td>ns</td>
<td>serial/single task jobs only</td>
</tr>
<tr>
<td></td>
<td>np</td>
<td>all parallel jobs, where all CPUs in 1 node should be requested and used</td>
</tr>
</tbody>
</table>

Table 8: LoadLeveler job classes on ecgate and c1a. Please note that parallel jobs are not permitted on ecgate. Use the `llclass` command on the relevant system for more details about each class.

Since the characteristics of the classes may be modified at any time in response to system changes and user requirements, they are not reproduced here. If you want more details, then the command `llclass` gives you a summary of the defined class names. Display the characteristics of a particular class by using the command:

```
llclass -l classname
```
#@ shell = /usr/bin/ksh
    # Specifies the shell that parses the script. If not
    # specified, your login shell will be used.
#@ initialdir = /scratch/ms/gid/uid
    # Specifies the path name of the directory to be used as the
    # initial working directory during the execution of the job. If
    # none is specified, the initial directory will be set to $SCRATCH.
#@ class = express
    # Specifies that your job should be run in the class (queue)
    # express.
#@ job_name = f77compile1
    # Assigns the specified name to the request
#@ output = ${job_name}.${host}.${jobid}.out
    # Specifies the name and location of STDOUT. If not given, the
    # default is /dev/null. By default the file will be written to
    # the directory specified by initialdir.
#@ error = ${job_name}.${host}.${jobid}.out
    # Specifies the name and location of STDERR. If not given, the
    # default is /dev/null. By default the file will be written to
    # the directory specified by initialdir.
#@ environment = COPY_ALL
    # Specifies that all environment variables defined in your shell
    # should be used. You can also list individual variables which
    # should be separated with semicolons.
#@ notification = error
    # Specifies that email should be sent in case the job fails.
    # Other options include always, complete, start, and
    # never. The default is notification = complete.
#@ job_cpu_limit = 00:01:00,00:00:55
    # Specifies the total CPU time which can be used by all
    # processes of a serial job step. In this job the hard limit
    # is set to 1 min and the soft limit to 55 sec. Note: All
    # limits are capped by those specified in the class.
#@ wall_clock_limit = 00:05:00,00:04:50
    # Specifies that your job requires HH:MM:SS of wall clock
    # time.
#@ queue
    # The queue statement marks the end of your LoadLeveler
    # keyword definitions and places your job in the queue. At least
    # one queue statement is mandatory. It must be the last keyword
    # specified. Any keywords placed after this in the script are
    # ignored by the current job step.

Figure 3: Job script example for ecgate.
9.1 Job submission

#@ shell = /usr/bin/ksh # Specifies the shell that parses the script. If not specified, your login shell will be used.

#@ class = np # Specifies that your job should be run in the class (queue) np, which is used to run parallel jobs.

#@ job_type = parallel # Specifies that your job is parallel

#@ job_name = pi-mpiomp # Assigns the specified name to the request

#@ output = $(job_name).$(schedd_host).$(jobid).out # Specifies the name and location of STDOUT. If not given, the default is /dev/null. The file will be written in the submitting directory, by default.

#@ error = $(job_name).$(schedd_host).$(jobid).out # Specifies the name and location of STDERR. If not given, the default is /dev/null. The file will be written in the submitting directory, by default.

#@ notification = error # Specifies that an email should be sent only if the job fails. Other options include: always, complete, start, and never. The default is notification = complete.

#@ resources = ConsumableCpus(32) ConsumableMemory(400mb) # Specifies quantities of the resources which will be ‘consumed’ by each process/task of a job step. ConsumableCpus and ConsumableMemory must be specified with a value which is greater than zero. The value ConsumableCpus specifies the number of threads per (MPI) process/task. This job will be run using 32 OpenMP threads per MPI process/task. The value ConsumableMemory specifies the (real) memory per (MPI) process/task which is shared between threads. This job will request 400 megabytes of real memory.

#@ node = 2 # Specifies that your job requires 2 nodes

#@ tasks_per_node = 2 # Specifies the number of (MPI) processes/tasks of a parallel job you want to run per node. In this example your job requires 2 processes/tasks per node.

#@ environment = OMP_NUM_THREADS = 32 # Specifies the number of OpenMP threads per (MPI) process/task by defining the environment variable OMP_NUM_THREADS.

#@ cpu_limit = 00:04:00 # Specifies the maximum CPU time in HH:MM:SS used by any single process. In this job 1 (MPI) process/task can use up to a total of 4 min CPU time. Note: All limits are capped by those specified in the class.

#@ wall_clock_limit = 00:03:00 # Specifies that your job requires HH:MM:SS of wall clock time. This job requires 3 min of wall clock time. Note: All limits are capped by those specified in the class.

#@ queue # The queue statement marks the end of your LoadLeveler keyword definitions and places your job in the queue. There must be at least one queue statement. It must be the last keyword specified.

Figure 4: Job script example for HPCF.
To submit a job script called jobname.cmd on ecgate, execute the command

```
llsubmit jobname.cmd
```

To submit a job script called jobname.cmd (in your c1a home directory) to the HPCF either execute the command

```
llsubmit jobname.cmd
```
on the interactive node on c1a or execute

```
rsh c1a-batch llsubmit jobname.cmd
```
on ecgate.

After you have submitted a job, you should monitor whether it is running as expected. The llq command will return a list of job steps in the LoadLeveler queues. To get a long status listing for a particular job use

```
llq -l job_id
```

If you use the option -s instead, the listing will also include some information on why a selected job remains in the NotQueued, Idle or Deferred state. To delete your own batch jobs from the LoadLeveler classes use

```
llcancel job_id
```
on the relevant system.

### 9.2 Multi-processor jobs

The HPCF clusters consist of 32 CPU Symmetric Multi-Processors (SMP) nodes. Please be aware that these systems are SMT enabled, which means that one processor can run 2 threads concurrently and therefore the nodes appear to have 64 (virtual) CPUs rather than 32 physical processors.

The recommended programming models on the HPCF are message passing using MPI (Message Passing Interface) and the shared memory model OpenMP. Applications using these models should port easily to other SMP clusters. While MPI enables communication between distributed nodes (memory), OpenMP is an efficient parallelisation technique within SMP nodes.

The most efficient approach – MPI, OpenMP or a mixture of both – needs to be assessed for each application. Mixed mode programming offers huge potential for efficiently performing code.

Whichever technique you use to run your code in parallel, please remember the following:

- the HPCF clusters are “supercomputers” and should be used as such
- avoid unnecessary idle time by e.g. not doing big MARS requests in parallel job steps
- use as few nodes as possible
- use as many CPUs within a node as possible

To convert a single processor job to use more than one processor/node requires some work. This is beyond the scope of this introductory guide. For some initial information, please refer to the links given at

[www.ecmwf.int/services/computing/hpcf/](http://www.ecmwf.int/services/computing/hpcf/)
9.3 Examples

Sets of example batch jobs for the ecgate server and the HPCF have been produced, covering the most common requirements. Copy them as necessary and use them to construct your own jobs; see the following link:

www.ecmwf.int/services/computing/job_examples/

9.4 Submission of time-critical jobs

This system allows users to automatically submit jobs to be run when certain points in the daily ECMWF operational forecast suite have been reached. The main purpose is to ensure that certain data is available before e.g. submitting a MARS request. This facility is using the ECaccess environment. It is available either through the Web interface of ECaccess or with the ECtools, available on ecgate or installed locally. For more information please refer to

www.ecmwf.int/services/computing/docs/tc_apps/tc_opt1.html

9.5 SMS

The Supervisor Monitor Scheduler (SMS) is a general-purpose application designed to schedule a large number of processes. It has been used for more than two decades to run ECMWF’s Meteorological Operational System (EMOS). SMS enables users to run a large number of programs which may have chronological dependencies and on one another, in a controlled environment with reasonable tolerance of both hardware and software failures, combined with good restart capabilities. If you would like to know more about SMS or to use it either on ecgate or at your site, please refer to www.ecmwf.int/publications/manuals/sms/ or contact User Support.

10 Fortran

The IBM XL Fortran compiler is available on both ecgate and the HPCF. It is an ANSI standard Fortran 90 compiler, but it can also handle Fortran 77 and Fortran 95 code, as well as most “industry extensions”.

Note the following:

- use the ecgate server for all non-parallel, low-performance or I/O intensive work (e.g. compilations, data retrieval, decoding/interpolation of data);
- use the HPCF preferably for parallel, well performing, computing intensive work;
- the same standard binary representations of data (e.g. IEEE, GRIB, BUFR) can be used on both platforms;
- the same major subroutine libraries (EMOSLIB, ECLIB, NAG) are available on both platforms.

This section gives enough information on the compiler and loader to allow you to run some simple jobs. A set of example jobs for each platform, to learn from or to construct your own jobs from, can be found at
www.ecmwf.int/services/computing/job_examples/

For full details see the IBM manuals or other documentation available from

www.ecmwf.int/services/computing/ecgate/ or
www.ecmwf.int/services/computing/hpcf/

or the relevant man pages.

Note: Ecgate uses a 32-bit application environment, as opposed to the HPCF, where a 64-bit application environment (i.e. 64-bit addressing mode) is the default.

10.1 Compiler invocation

The basic IBM Fortran compiler/linker is called xlf. To cater for various needs, e.g. thread safe compilation, several variants of the xlf command, which only affect the default compiler options (stanzas), are available. The supported language for all these variants is the same full Fortran 90/95 standard. The general command is

```
mpxlfnnth
```

where

- `mp` is an optional prefix needed when using MPI extensions
- `nn` is an empty string or "90" or "95" (FORTRAN 90 or 95)
- `th` is an empty string or ".r" (POSIX thread safe) or ".r7" (draft 7 POSIX thread safe)

In addition to these IBM specific names, the command f77 is also available.

This probably gives you too many commands to choose from. For most users’ needs the commands xlf for non-parallel, xlf_r for all programs using OpenMP and mpxlf_r for all programs using MPI might be sufficient. This implies that on ecgate, where you are expected to run serial jobs only, the command xlf or xlf90 should suffice. The actual settings for each compiler invocation are given in the configuration file /etc/xlf.cfg.

It is recommended that you use the xlf/xlf90 command also to link object files, possibly with libraries, to create an executable.

In general, the environments on ecgate and HPCF are not identical. To check which software is actually installed, you can use the lslpp command, e.g.

```
lslpp -1|grep xlf
```

Note: The following non-standard default compiler options are set:

- on ecgate: -qextname
- on HPCF: -qextname -q64

where -q64 will enable 64-bit addressing-mode and -qextname will add an underscore "." to the name of library routines called. All libraries provided by ECMWF have been compiled using these default options.
10.2 Compiler options

There are three methods, in order of ascending precedence, to set compiler options:

1. keep your own version of the xlf configuration file (see /etc/xlf.cfg for a configuration file example) and run the compiler with an appropriate -F option (see XL Fortran User’s Guide)

2. specify options on the command line, e.g.

   xlf -c -qautodbl=dbl4 -O prog.f

3. include options as directives in your source code. These options will start with the label @PROCESS, as in:

   @PROCESS autodbl(dbl4)
   subroutine sub(a,b)
   ...
   return
   end

   The scope of these directives is limited to the compilation unit following the directive. These directives are case insensitive and need to be placed before the first statement in the source code.

To see what files and commands are involved or what options are in effect for each step use the compiler option -V.

A classified listing of the most common compiler options is given in the following subsections.

10.2.1 Standard options

Some of the most common xlf/xlf90 compiler options are summarized in table 9.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c</td>
<td>compilation only</td>
</tr>
<tr>
<td>-g</td>
<td>produces information for symbolic debugging</td>
</tr>
<tr>
<td>-I &lt;dir&gt;</td>
<td>adds directory named to search for include files or module files</td>
</tr>
<tr>
<td>-qmoddir=&lt;dir&gt;</td>
<td>puts the module files in named directory</td>
</tr>
<tr>
<td>-l&lt;key&gt;</td>
<td>at linking, searches for external entries in the library with a name matching lib&lt;key&gt;.a or lib&lt;key&gt;.so</td>
</tr>
<tr>
<td>-L &lt;dir&gt;</td>
<td>searches also in named directory for library files</td>
</tr>
<tr>
<td>-o &lt;name&gt;</td>
<td>specifies the name of the output file produced, instead of the default name, e.g. a.out for an executable</td>
</tr>
<tr>
<td>-p</td>
<td>requests basic profiling</td>
</tr>
<tr>
<td>-pg</td>
<td>requests extended profiling</td>
</tr>
</tbody>
</table>

Table 9: Common xlf compiler options.

An example:

   xlf90 -o myprog -g -p myprog.f

will compile the file called myprog.f and create an executable file called myprog, with debugging information included and basic profiling activated.
10.2.2 Options that control Fortran source format

Commonly used Fortran source file suffixes are .f, .F, .f90, .F90, .f95 and .F95. The different stanzas of the xlf compiler will only understand specific file suffixes and also assume that the source code is in a specific format, free or fixed, as can be seen from the table below.

<table>
<thead>
<tr>
<th>Command</th>
<th>Suffix</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>xlf</td>
<td>.f (.F)</td>
<td>fixed</td>
</tr>
<tr>
<td>xlf90</td>
<td>.f (.F)</td>
<td>free</td>
</tr>
<tr>
<td>xlf95</td>
<td>.f (.F)</td>
<td>free</td>
</tr>
</tbody>
</table>

The options available to change these default rules on source code format are listed in table 10.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-qfixed[=&lt;right-margin&gt;</td>
<td>fixed source format plus optional maximum line length</td>
</tr>
<tr>
<td>-qfree</td>
<td>free source format</td>
</tr>
<tr>
<td>-qsuffix=f=&lt;suffix&gt;</td>
<td>defines the suffix of source file to &lt;suffix&gt;</td>
</tr>
<tr>
<td>-qsuffix=cpp=&lt;suffix&gt;</td>
<td>defines the suffix of source file for preprocessor</td>
</tr>
</tbody>
</table>

Table 10: xlf options to control Fortran source format.

For instance,

```
xlf90 -qfixed a.f90 b.F90 -qsuffix=f=f90 -qsuffix=cpp=F90
```

will compile the two files a.f90 and b.F90 in fixed source code format. The compiler will be invoked for files with suffix .f90 and the C preprocessor will first be invoked for files with suffix .F90.

If you only want to run the C preprocessor and keep the file with source code, you should use the options “-d -noobject”.

Also note that the different stanzas of the Fortran compiler, xlf, xlf90 or xlf95, will not check whether the code complies with the Fortran 77, Fortran 90 or Fortran 95 standard, respectively. The option -qlanglvl=<level> will allow you to check for non-conformance to a specific Fortran standard. By default <level> is set to ‘extended’, which is equivalent to no language level checking. This option can be combined with the option -qhalt=1 to stop compilation, if non-conformant source code is used, e.g.

```
xlf90 -qsuffix=f=f90 -qlanglvl=f90std -qhalt=1 myprog.f90
```

will compile the program myprog.f90 and check for compliance of the code to the Fortran 90 standard.

10.2.3 Listings options

The options given here are especially useful during porting to the AIX systems. The listing file generated will have a file suffix .lst and will contain different listing sections requested with one of the options given in table 11.

10.2.4 Compatibility options

The options given in table 12 are useful when porting some code from other platforms.

Example:
### 10.2 Compiler options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-qsource</code></td>
<td>includes source code section</td>
</tr>
<tr>
<td><code>-qlist</code></td>
<td>includes object section (sort of assembler language)</td>
</tr>
<tr>
<td><code>-qlistopt</code></td>
<td>includes options sections</td>
</tr>
<tr>
<td><code>-qref</code></td>
<td>includes cross reference section</td>
</tr>
<tr>
<td><code>-qreport</code></td>
<td>includes a section on optimization done by the compiler</td>
</tr>
<tr>
<td><code>-V</code></td>
<td>generates information on the progress of the compilation</td>
</tr>
</tbody>
</table>

**Table 11: xlf listings options.**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-qautodbl=&lt;setting&gt;</code></td>
<td>automatically converts the precision of floating point calculations to a higher level, e.g. suboption <code>dbl4</code> promotes floating-point objects that are single-precision (4 bytes) or that are composed of such objects (e.g. COMPLEX or array objects): REAL(4) is promoted to REAL(8), COMPLEX(4) is promoted to COMPLEX(8); default is none</td>
</tr>
<tr>
<td><code>-qdpc</code></td>
<td>increases the precision of real constants</td>
</tr>
<tr>
<td><code>-qrealsize=&lt;n&gt;</code></td>
<td>sets the size of default real variables to n bytes; default is 4</td>
</tr>
<tr>
<td><code>-qintsize=&lt;n&gt;</code></td>
<td>sets the size of default integer and logical variables to n bytes; default is 4</td>
</tr>
<tr>
<td>`-q[save</td>
<td>nosave]`</td>
</tr>
<tr>
<td>`-q32</td>
<td>-q64`</td>
</tr>
<tr>
<td>`-qextname</td>
<td>-qnoextname`</td>
</tr>
</tbody>
</table>

**Table 12: xlf compatibility options.**

```bash
xlf90 -qautodbl=dbl4 -qdpc myprog.f
```

will increase the precision of all single precision floating point calculations to double precision calculations. Real constants are also promoted to double precision constants.

#### 10.2.5 Optimization options

Many options and sub-options are available to help you reduce the execution time of your program (see table 13).

For `-qarch` and `-qcache`, the sub-option `auto` will automatically select the appropriate instruction set and cache usage for the architecture on which the compiler is run.

It is recommended to start with a relatively low optimization level, before going to more aggressive optimization. For example

```bash
xlf90 -O3 -qstrict -qarch=auto -qtune=auto -qnohot myprog.f
```

will achieve a very good level of optimization, and the results will not differ from those produced without optimization.
Option | Description
--- | ---
-O[2|3|4|5] | optimization of source code at increasing levels, -O3, -O4 and -O5 are more aggressive and may change results; default is no optimization
-qhot | performs high order transformations, replacing some code by high performance library calls (implied by -O3)
-qipa | does interprocedure analysis
-qstrict | makes sure that optimization with options like -O3, -qhot and -qipa does not alter results
-Q | performs inlining; at least optimization -O2 is required; default is none
-qarch=<arch> | allows the compiler to take maximum advantage of the machine instructions specific to an architecture
-qtune=<arch> | tunes instruction selection, scheduling, and other architecture-dependent performance enhancements to run best on specified architecture
-qcache=<arch> | tunes cache usage to specified architecture

Table 13: xlf Version 12 optimization options.

### 10.2.6 Debugging options

The options given in table 14 will help you in your first attempts to debug your programs.

Option | Description
--- | ---
-C | check array element references
-qextchk | check procedure argument types, common block types and module data types across routines
-qundef | inhibits implicit typing of variables
-qltttrap=|suboption | determines what type of floating-point exceptions to detect (not recommended, see section 10.4)
-qinitauto=<value> | initializes automatic variables to a specific value

Table 14: xlf debugging options.

For instance,

```
xlf90 -qinitauto=FF -qltttrap=invalid:en myprog.f
```

will abort the program execution, if some variables are not initialized before being used.

### 10.2.7 Linker options

Options relevant to the link editor (ld command), usually starting with -b, will be ignored by the Fortran compiler and will be passed to the ld command. There are numerous options to modify the behavior of the link editor. In table 15 a few commonly used options are given. The complete list of options and a detailed description of the ld command are available from the man page.
## 10.3 Input/output

A file can be connected to a Fortran unit number inside or outside the program. To internally connect a unit number to a data file, use the `FILE=` option on the `OPEN` statement e.g.

```fortran
OPEN (UNIT=20, FILE='myfile', ...)
```

where `myfile` is a file in your current working directory or a full path name. If the above `OPEN` statement is not used, a `WRITE(20, ...)` statement writes by default to a file `fort.20` (without leading zero, e.g. `fort.3` for unit number 3). Valid unit numbers go from 0 to 2147483647 where numbers 5, 6, and 0, are by default connected to stdin, stdout, and stderr respectively. They can be redirected to any named files using the redirection facility outside the program e.g.

```
a.out < infile > outfile
```

In this example unit 5 is connected to `infile` and unit 6 to `outfile`. If the unit number is omitted or specified by an asterisk `*` then unit 5 is used for reading and unit 6 for writing, which are again connected to stdin and stdout respectively.

There are two more methods to connect a file to a Fortran unit number outside the program. Use the Unix `ln` command to link your file to the standard file name, e.g. the command

```bash
ln -s myfile fort.20
```

given before a program execution will insure that the statement `WRITE(20, ...)` writes to `myfile`.

A second method uses environment variables `XLFUNIT_nn` for file connection, where the unit number `nn` must not have any leading zeros. Also you must set the run-time option `unit_vars` to use this method. The (ksh) commands

```bash
export XLFRTEOPTS="unit_vars=yes"
export XLFUNIT_20="myfile"
```

executed before a program execution will ensure that the statement `WRITE(20, ...)` writes to `myfile`.

For more details see the "XL Fortran for AIX Users’ Guide".

---

### Table 15: xlf linking options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`-b32</td>
<td>-b64`</td>
</tr>
<tr>
<td><code>-bmaxdata=N</code></td>
<td>sets the maximum size for the user data area</td>
</tr>
<tr>
<td><code>-bmaxstack=N</code></td>
<td>sets the maximum size for the user stack area.</td>
</tr>
<tr>
<td>`-bshared</td>
<td>dynamic`</td>
</tr>
<tr>
<td><code>-bstatic</code></td>
<td>shared objects will be statically linked</td>
</tr>
<tr>
<td><code>-bloadmap:&lt;file&gt;</code></td>
<td>writes a report on the linking command to file; approximately equivalent to the <code>-bnoquiet</code> option</td>
</tr>
<tr>
<td><code>-bmap:&lt;file&gt;</code></td>
<td>writes an address map of all symbols included to file</td>
</tr>
</tbody>
</table>

---

February 12, 2010 29
10.4 Debugging

There are various facilities available to help to debug Fortran jobs. If possible, start with simple techniques to track down errors. Some points to remember:

- produce some source code listing (see table 11)
- if you use PRINT statements to track down bugs, this can produce a lot of output, so redirect the stdout/stderr to a file
- try inserting IF statements that verify assumptions
- try to use 'debugging' compiler options (see table 14)
- the optimizer can, and very often does, change the order of execution of program statements
- the optimizer may change the output of a program, if the program violates the Fortran language specification

On both ecgate and the HPCF Floating Point Exceptions (FPE) are not trapped by default. As the IBM FPE trapping is quite expensive (option -qflttrap), the use of the signal_trap routine available in ECLIB is recommended instead. This routine can be used by adding the following code to your program and linking to ECLIB.

```fortran
INTEGER*4 CORE_DUMP_FLAG, IRETURN, SIGNALS(1), SIGNAL_TRAP
CORE_DUMP_FLAG = 0
SIGNALS(1) = 0

IRETURN = SIGNAL_TRAP(CORE_DUMP_FLAG, SIGNALS)
```

Signals trapped by default are SIGFPE, SIGILL, SIGBUS, SIGSEGV and SIGXCPU.

If the basic techniques mentioned above don’t work for you, there are also two debuggers available: the command-line debugger dbx with its parallel variant pdbx and the graphical debugger Totalview. However, their use is rather complex and resource consuming. For both debuggers you have to compile and link your code with the option -g and should switch off all optimization.

Before you start Totalview in your batch job, you have to set the DISPLAY environment variable. Then, if you execute a parallel program, you call poe under the control of Totalview and pass the executable file name etc. using the Totalview -a flag:

```bash
totalview [search_path <dir>] poe -a ./prog [my_args]
```

You can also specify a search path for source files. For more information please refer to the Totalview on-line manuals or directly to the Totalview Technologies site (www.totalviewtech.com).

The dbx/pdbx debugger is called with the executable’s name as a parameter

```bash
pdbx ./prog
```

For more information please refer to the pdx/pdbx man page and to ‘IBM Parallel Environment for AIX, Operation and Use, Volume 2’. To examine core files, please use dbx.
10.5 Porting hints

Here we list some hints that could help you to port your codes to the IBM platforms ecgate and HPCF.

- Non-standard Fortran code may produce different results on different platforms. If you don’t get the same results on the IBM systems as on another system, you can use the compiler option `-langlvl` to check for source code non compliant with a defined Fortran standard.

- Don’t forget that floating point exceptions are not fatal by default. Use the `signal_trap` routine to intercept them.

- When starting your optimization exercise, start with `-O2` or `-O3` optimization level, use the profiling tools and always check your results.

- Note that the new Fortran NAMELIST I/O format

```
&INFO
DATE=20020805
/
```

is expected by default. You can still use the old namelist format by setting the runtime environment variable `XLFRTEOPTS` to `NAMELIST=old`.

10.6 Optimization

Even though performance is of less concern on the server, you should always aim at optimizing your programs and job requests. This will increase the performance of your programs and also reduce the general load on the machine, hence benefiting everyone.

The main area for optimization is usually the CPU usage, whereas I/O performance is generally less crucial. Concentrate all your optimization efforts on the expensive parts of your code, which can be identified using the tools mentioned in section 10.7.

Before you embark upon any hand tuning, you should try to use optimized libraries (e.g. MASS, ESSL, NAG, see section 11) and the compiler’s optimizer (see table 13). However, you should note that optimization generally involves code reordering, which might produce different results. If at this stage specific routines fail, they can continue to be compiled at a lower level of optimization by using source embedded compiler directives.

When coding I/O you should take note of the HPCF `$TEM` disk block size of 4 MB. The best I/O transfer rates are obtained with sequential access and record sizes equal to or bigger than the disk block size.

10.7 Profiling and timing

Before starting any optimization it is important to know how a program behaves and, if it behaves badly, where it does so. There are several tools on both platforms to help with this task. They are described below.

To get a basic idea of the overall performance of your job, look at the output produced by `eoj`. It gives basic statistics about your job, such as efficiency, load balance, accounting, etc. On the HPCF this utility has been included in the system epilogues but can be used interactively at any time, also on ecgate to check on the progress of your executing job. To do this, type
Introduction to computing facilities

eoj job_ID

on the relevant system (use $lq$ to find the job ID of your job). See Figure 5 for an example output.

Profiling a program allows you to find out which parts of the code are most active, so that you can concentrate your tuning efforts on these areas. In addition timing information can be obtained on entire executables or specifically between user defined check points.

The IBM tool Xprofiler is available to profile your application. If you have problems running this X-application or have reservations about GUI tools, you can also use the common (g)prof command, which is available on most Unix platforms. In both cases you need to compile and link with additional compiler options and run your application with different test cases for at least a few seconds. If the application terminates normally, it will create a file (g)mon.out or, in the case of a parallel run, several output files (g)mon.out.i, where i is the task id. You then analyze the created (g)mon.out files with one of the available profiling tools.

Note the following (g)mon.out file issues:

- each run of your program will create new gmon.out files (overwriting any existing gmon.out files) so rename old gmon.out files first, if you want to keep them
- recompling the program invalidates all older gmon.out files; saving the old binary before recompiling will keep older gmon.out files valid

Xprofiler

The IBM Xprofiler (part of the IBM Parallel Environment, POE) is a GUI tool to analyze the performance of serial and parallel applications. It provides a visual indication of the call tree and the most active routines.

To enable some useful Xprofiler features (e.g. display included functions and source statement profiling) the application should be compiled and linked with the option $-g$ (and $-qfullpath$, if necessary) in addition to $-pg$ (and usual optimization options).

To summarize some important points:

- compile and link with $-g$ $-pg$ (and usual optimization options)
  
  $xlf -g -pg -O3 prog.f -o grog$
- execute program against chosen test case(s)
- invoke Xprofiler on the executable and the gmon.out file
  
  $xprofiler prog gmon.out*$
- graphical indication of call tree provided
- visual indication of most active routines
- click on routine to get FORTRAN statement level profiling
- collapse/hide library information that isn’t needed (like system libs)
- uncluster functions and zoom-in to see important details

More information can be obtained from the integrated help tool but also from the man page.

Gprof

The tool gprof provides nearly the same call-graph based information as Xprofiler but as ascii output only. To use gprof compile and link with the compiler options $-g$ $-pg$. Then run your application and analyze the created gmon.out file(s) by starting gprof as
Run at Fri Sep 15 05:58:08 GMT 2006 on hpce1004 for jobstep hpce0204.172746.0
Queued : Fri Sep 15 05:15:11 GMT 2006 for 12 seconds
Dispatched : Fri Sep 15 05:15:23 GMT 2006 for 2565 seconds
Exit Code : 0
Job Name : modeleps_01
Step Name : step_1
Owner : emos
Notify User : emos@hpce0205
Unix Group : ma
Account : oosu
STDIN : /dev/null
STDOUT : /emos_dir/0001/log/mc/main/00/fc/pf/legA/01/modeleps.1
STDERR : /emos_dir/0001/log/mc/main/00/fc/pf/legA/01/modeleps.1
Class : op
Step Type : General Parallel
Node Usage : not_shared
Step Cpus : 96
Total Tasks :
Blocking :
Node actual : 3
Adapter Req. : (csss,MPI,US,not_shared,HIGH,instances=1,)
Resources : ConsumableCpus(4) ConsumableMemory(3.125 gb)
Comment : (ec_smt=yes)

#*#* Next 3 times NOT up-to-date (TOTAL CPU TIME given later IS accurate)
Step User Time : 1+05:00:35.753867
Step System Time : 00:03:33.974689
Step Total Time : 1+05:04:09.728556 (104649.728556 secs)

#*#* Last 3 times NOT up-to-date (TOTAL CPU TIME given later IS accurate)
Context switches : involuntary = 2162623, voluntary = 19690547
per second = 843 7676
Page faults : with I/O = 2978, without I/O = 103107
per second = 1 40

------------------<----------- CPU ------------->-<------------ MEMORY ---------------->------------
n a h s r k d | total average current | total current maximum ratio | physical consumed consumed | alloc'd consumed consumed maximum | CPU physCpu/ physCpu/ | memory memory memory memory/ | seconds alloc'd alloc'd | on node on node on node alloc'd |
Node ?. s s | consumed physCpu physCpu | [MB] /alloc'd [MB] memory | task list
-----------------|-------------------------------|--------------------------------------|-----------
hpce1004 M 8 4 | 37796.72 92% 2% | 25600 0% 6591 26% | 0:1:2:3... hpce1005 . 8 4 | 37916.61 92% 0% | 25600 0% 6619 26% | 8:8:10:...
hpce1006 . 8 4 | 37592.18 91% 0% | 25600 0% 6532 26% | 16:17:1...

Min | 37592.18 | 6533 26% | Max | 37916.61 | 6619 26% |

wlmnpghl: 1687411/(6591) wlmnpghl: 1677448/(6552)
Elapsed: 2565.00 sec Alloc’d memory/task: 3200 MB
CPU Tot: 113305.52 sec { 1+07:28:25} Average: 37769 s/node, 4721 s/task

System Billing Units used by this jobstep = 446.558

Figure 5: A typical eoj output for a 96 processor job on the HPCF.
gprof prog gmon.out*

If you pass more than one gmon.out file to gprof, it will report on the sum of the profiling information. As the output can become quite large, you might want to redirect it into a file:

```
gprof prog gmon.out.0 gmon.out.1 > myprof
```

**Prof**

The tool prof provides a flat profile, i.e. time spent in called routines is not counted as time spent in the parent routine. To use prof, compile and link with the `-p` option. Then run your application and analyze the created mon.out(.i) files by starting prof as

```
prof prog -m mon.out*
```

**Obtaining timing information**

An easy way to obtain basic timing information on an entire application is to use the AIX `timex` command, which reports on ‘Real’, ‘User’ and ‘System’ time, where the total CPU time is the sum of ‘User’ and ‘System’ time. To use this command, just put it in front of your executable

```
timex prog
```

For more information refer to the `timex` man page.

If you want to time between specific lines of your code, the functions `MCLOCK()` and `RTC()` are available. The integer function `MCLOCK()` returns CPU time in 1/100th of seconds (AIX restriction). The real function `RTC()` is a real time clock which returns seconds correct to microseconds.

If you want to use `MCLOCK()` the timed code must run significantly longer than 1/100th sec. If your code does not fulfill this requirement, you might want to wrap it into a ‘time multiplier’ loop. However, be careful as this may wrongly hide “cache miss” effects. The solution is to use `RTC()` and run CPU-bound on an otherwise quiet system.

The following examples show how to use both functions:

```
implicit (none)
real*8 r0,rtc, cpu_secs, real_secs
integer m0, mclock
...
r0= rtc()
m0= mclock()
...
>code you want to time<
...
cpu_secs=( mclock()- m0) * 1.d-2
real_secs= rtc()- r0
```

You must also be aware that the optimizer may move calls to `MCLOCK()` and `RTC()` or other bits of your code. To avoid this, use the compiler option `-qstrict` or insert additional print statements.

Furthermore, on HPCF some libraries are provided for low overhead elapsed-time measurement of MPI routines. They are making use of the AIX hardware performance monitor (HPM). See `/usr/local/lib/trace/README` on HPCF for more information.
11 Software libraries

This section describes how to use software libraries provided by ECMWF and third parties, as well as how to create your own libraries.

11.1 General software libraries

General software libraries are those libraries which contain commonly used routines of general interest. Unless stated otherwise, these libraries are made available in binary form on both ecgate and the HPCF for direct linking to your program. For more information please refer to the library page at

[www.ecmwf.int/services/computing/docs/libraries/](www.ecmwf.int/services/computing/docs/libraries/)

**EMOSLIB**

a library of those routines used within ECMWF’s operational forecast suite thought to be of more general interest (e.g. the Interpolation package, BUFR software, GRIBEX, etc);

**ECLIB**

a collection of routines of general interest (e.g. date manipulation) put together by ECMWF User Support;

**NAG**

a commercial library of mathematical routines from the Numerical Algorithms Group Ltd (NAG), Oxford;

**(P)ESSL**

IBM Engineering and Scientific Subroutine Library, a Fortran (and C/C++) callable library of optimized mathematical subroutines from IBM for AIX platforms only. (P)ESSL includes most of the BLAS routines and can therefore be linked to, instead of BLAS, without problems. (P)ESSL also contains equivalents of most LAPACK routines. As the subroutines in this library are IBM specific, your code becomes less portable. However, these routines are generally optimized better than either those in other libraries or your own code.

The serial version is suitable for use on one node and the parallel version is needed for use on more than one node. An SMP version is also available for e.g. OpenMP programs. See the following examples:

```
xlf -O prog.f -lessl
```

will compile and link a program using ESSL, to run on one single CPU.

```
xlf_r -O -qnosave prog.f -lesslsmmp
```

will compile and link an OpenMP program using ESSL, to run on multiple CPUs within one node.

```
mpxlf_r -O prog.f -lesslsmmp -lpesslsmmp -lblacssmp
```

will compile and link a program using OpenMP and MPI, i.e. a program combining Shared Memory Programming (within one node) and Distributed Memory Programming (across nodes).

As there are no man pages available, please refer to the IBM manuals for further information.
MASS

IBM Mathematical Acceleration SubSystem, contains optimized versions of commonly used Fortran intrinsic functions. It is available on AIX platforms only. Vector and scalar versions of MASS are available. The increased performance of MASS, compared to the standard mathematical library, is at the expense of an insignificant amount of accuracy.

The scalar version of MASS can be accessed with the compiling/linking options

- `L/usr/local/lib/mass` -lmass

and the vector version (which requires code changes) with the option

- `L/usr/local/lib/mass` -lmassv

For more information, please refer to the IBM documentation at [www.ecmwf.int/publications/manuals/hpcf/](http://www.ecmwf.int/publications/manuals/hpcf/)

To use routines from any of these libraries, you must specify them at the linker/loader stage. Some of the library names are held in environment variables:

- `$EMOSLIB` # for EMOSLIB
- `$ECLIB` # for ECLIB
- `$NAGLIB` # for the NAG library

To specify one or more to the loader, add the library variable after your program name e.g.

```bash
xlf prog.o -lessl $EMOSLIB # link with ESSL and EMOSLIB
```

**Note:** The ECMWF libraries EMOSLIB and ECLIB are non-parallel, thread-safe versions compiled in 32- and 64-bit addressing mode on ecgate and HPCF, respectively.

### 11.2 Creating and using your own library

**Creating a library**

Use `ar` to create your own static archive library, e.g.

```bash
ar -rv libmy.a *.o
```

Use `ar` to build static libraries, but `ld` to build shared libraries.

**Referencing a library**

Libraries are referenced as part of the compiler/linker command. Thus:

```bash
xlf prog.o -lllibrary
```

where `liblibrary.a` is the full name of the relevant static library.
12 Graphics

Graphics facilities at ECMWF are available on the workstation/server systems. There are no graphics facilities on the HPCF.

The two principal graphics packages, developed in-house, are the METVIEW visualization system and the MAGICS Fortran library. Metview is an interactive visualization system that has been specifically designed for use by research meteorologists. It can also be used in batch mode for repetitive tasks. MAGICS is a Fortran callable library of graphics routines intended for those who have to do their own graphics programming. It is heavily used within Metview. Both packages are available on the ecgate server and locally in some Member States.

In addition to METVIEW and MAGICS, some commercial graphics packages and some public domain packages are available on an 'as-is' basis. For detailed information on all the available graphics software, see the graphics online information pages at

www.ecmwf.int/services/computing/docs/graphics/