UNIX システム利用入門

システム運用掛　宮 嵜　　洋
佐々木 一孝

1.　システム概要

1.1.　ハードウェア

本センターにおける UNIX システム†は 2 台のスーパーコンピューターシステムと運用支援システム及び各種サーバー群で構成されています。以下に主な構成を示します。

超並列型スーパーコンピューター　HITACHI SR8000/MPP

ノード数	144 ノード（1152 プロセッサー）
理論演算性能	ノード当り 14.4GFLOPS（プロセッサー当り 1.8GFLOPS）
主記憶容量	ノード当り 16GB
ホスト名	mpp-s.cc.u-tokyo.ac.jp、mpp-p.cc.u-tokyo.ac.jp （バッチシステム mpp-bt.cc.u-tokyo.ac.jp）

ベクトル並列型スーパーコンピューター　HITACHI SR8000/128

ノード数	128 ノード（1024 プロセッサー）
理論演算性能	ノード当り 8GFLOPS（プロセッサー当り 1GFLOPS）
主記憶容量	ノード当り 8GB
ホスト名	bulk-s.cc.u-tokyo.ac.jp、bulk-p.cc.u-tokyo.ac.jp （バッチシステム bulk-bt.cc.u-tokyo.ac.jp）

☞ SR8000/128 は定額負担金によるバルク利用システムとして運用

運用支援システムと主なサーバー群

運用支援システム	m-unix.cc.u-tokyo.ac.jp
X 端末制御用サーバー	x-controller1.cc.u-tokyo.ac.jp
入出力機器制御用サーバー	io-controller.cc.u-tokyo.ac.jp
ニュースサーバー	newsmachine.cc.u-tokyo.ac.jp
アプリケーションサーバー	mathematica.cc.u-tokyo.ac.jp macsyma.cc.u-tokyo.ac.jp

周辺機器

プリンター装置（スーパーコンピューティングニュース Vol.3 No.3,2001.5 参照）

ポストスクリプトプリンター（モノクロ）　2 台

高画質カラープリンター（用紙／ＯＨＰ）　2 台

磁気テープ装置（スーパーコンピューティングニュース Vol.3 No.4,2001.7 参照）

DDS（DAT）、QIC、8mm、1/2 インチオープンリール型、1/2 インチカートリッジ型

その他入出力機器

MO、CD-ROM、DVD-ROM

---

† 本稿は特に断らない限り、スーパーコンピューターSR8000/MPP、SR8000/128 の利用を前提として記述しています。

1.2. ソフトウェア

本センターのスーパーコンピューターは 1 つのシステムにインタラクティブ（TSS）環境とバッチ（BATCH）環境の両方を持っています。

SR8000/MPP（基本負担金方式）

mpp-bt (BATCH)	
mpp-s (TSS)	mpp-p (TSS)

SR8000/128（定額負担金方式）

bulk-bt (BATCH)	
bulk-s (TSS)	bulk-p (TSS)

TSS 環境とはログインして対話的に利用する環境で、主にプログラムの作成・編集、コンパイルやデバッグ実行、バッチジョブの投入などに使用します。バッチ環境とはバッチジョブ実行専用（ログイン不可）のシステムで、インタラクティブ環境から投入されたプログラム（ジョブ）を実行する環境です。本センターでは大規模バッチ環境と 2 種類のインタラクティブ環境をサブシステムという形態で用意しています。各サブシステム上で HI-UX/MPP と呼ばれる UNIX OS が独立して動作しているため、1 つのシステムで複数のホスト名を持っています。以下にサブシステムを簡単に紹介します。

インタラクティブ処理用サブシステム

mpp-s.cc.u-tokyo.ac.jp、bulk-s.cc.u-tokyo.ac.jp
スカラージョブを対話的に実行できます。主にプログラムの作成・編集、コンパイル、バッチジョブの投入に使用します。

mpp-p.cc.u-tokyo.ac.jp、bulk-p.cc.u-tokyo.ac.jp
要素並列ジョブおよび 2 ノードまでの並列実行を対話的に実行できます。主にテスト、デバッグ実行に使用します。（実際の計算はバッチジョブで実行して下さい。）

バッチ処理用サブシステム

mpp-bt.cc.u-tokyo.ac.jp、bulk-bt.cc.u-tokyo.ac.jp
スカラージョブ、要素並列ジョブ、拡張記憶（ES）使用ジョブ、16 ノードまでの並列実行に使用できます。（ログイン不可）

利用負担金については、各々のスーパーコンピューターシステム毎に体系が異なります。SR8000/MPP は基本負担金方式による利用者毎の従量課金ですが、SR8000/128 は利用者グループを一括登録するシステムで定額負担金方式（バルクコース）による定額課金です。使い方についてはほとんど共通ですが、性能やリソースの異なる別のシステムであり、利用申請書や登録方法も異なります。

☞ 定額負担金方式についてはスーパーコンピューティングニュース記事「定額負担金の『バルクコース』運用開始について」（Vol.3 No.1, 2001.1）、「利用負担金改正のお知らせ」（Vol.3 No.2, 2001.3）を御覧下さい。

本センターのスーパーコンピューターにインストールされている主なソフトウェアは以下の通りです。

言語プロセッサー（コンパイラー）
- 最適化 FORTRAN77
- 最適化 FORTRAN90
- 最適化 C
- 最適化 C++

並列化支援アプリケーション
- PARALLELWARE
- Parallel FORTRAN
- MPI、PVM

数値計算ライブラリー
- MATRIX/MPP
- MATRIX/MPP/SSS
- MSL2
- BLAS、LAPACK、ScaLAPACK（本誌別記事参照）

その他ライブラリー
- Gaussian98（スーパーコンピューティングニュース Vol.3 No.4, 2001.7 参照）

☞ ソフトウェアは SR8000/MPP と SR8000/128 とで同じです。最新のバージョン番号等はスーパーコンピューティングニュース「システム変更等のお知らせ」で御確認下さい。

以下は運用支援システム（m-unix）の主なソフトウェアです。

言語プロセッサー（コンパイラー）
- 最適化 FORTRAN77
- 最適化 PASCAL
- C
- LISP
- C++
- PROLOG

数値計算ライブラリー
- MATRIX/M
- MATRIX/M/SSS
- MSL2

その他ライブラリー
- GKS(C、F)
- DEQSOL/LIB
- Mathematica（センターニュース Vol.28 No.4, 1996.7 参照）
- Macsyma（センターニュース Vol.28 No.4, 1996.7 参照）

☞ UNIX システムには若干のフリーソフトウェアがインストールされていますが、本センターではフリーソフトウェアの新規導入及び保守、バージョンアップ等のサポートを行っておりません。また、障害等が発生した場合にはサービスを中止することがあります。

SR8000/MPP　128 ノードジョブ運用

SR8000/MPP では 2001 年 10 月より、128 ノードジョブ運用の試行サービスを開始する予定です。本サービスは原則として毎月第１週の週末にバッチシステムを拡大し、128 ノード（1024 プロセッサー）のバッチジョブを実行できるようジョブクラス P128 を設けるものです。詳細は本誌別記事「SR8000/MPP 128 ノードジョブ運用試行サービス開始について」を御覧下さい。

2. 接続方法

2.1 ネットワークからの接続

SSH(Secure Shell バージョン 1)または telnet による接続が可能です。接続先は以下を御覧下さい。IP アドレスは変更する場合がありますのでホスト名で御利用下さい。なお、SSH による接続方法はスーパーコンピューティングニュース Vol.2 No.4, 2000.7「UNIX システムへの SSH（Secure Shell）接続について」を参照して下さい。

IP ネットワーク経由時のホスト名（2001 年 5 月 1 日変更）

システム	OS	ホスト名	IP アドレス *1
運用支援システム	VOS3/FS	m-vos.cc.u-tokyo.ac.jp	130.69.240.40
	HI-OSF/1－MJ	m-unix.cc.u-tokyo.ac.jp	130.69.240.41
HITACHI SR8000/MPP	HI-UX/MPP	mpp-s.cc.u-tokyo.ac.jp *2	130.69.240.51
		mpp-p.cc.u-tokyo.ac.jp *3	130.69.240.52
HITACHI SR8000/128	HI-UX/MPP	bulk-s.cc.u-tokyo.ac.jp *2	130.69.240.56
		bulk-p.cc.u-tokyo.ac.jp *3	130.69.240.57

*1 IP アドレスは変更になる場合がありますので、極力ドメインネームサーバーを参照し、ホスト名でご利用ください。

*2 スカラージョブ処理用サブシステム。

*3 要素並列ジョブを含むノード占有ジョブ処理用サブシステム。

2.2. 電話回線からの接続

以下の電話番号で接続することができます。

電話回線利用 TSS の電話番号(2001 年 4 月 2 日変更)

通信速度(規格)	VOS3	UNIX	端末側設定	
			設定項目	設定内容
300 ～ 56000 bps *1 (V.90, K56 flex 他) (V.42, MNP 2 ～ 4, 10 V.42bis, MNP 5)	03-3815-6391		データー長	8 ビット
			パリティ	なし(NONE)
			ストップビット	1 ビット
			エコー	リモートエコー
			文字コード	UNIX は JIS コード VOS3 は JIS, SHIFT-JIS, EUC コードの選択が可

*1 速度の切り替えは端末側のモデムにあわせて自動的に行われる。

2.3. センター内端末からの接続

ユーザーフロアのX端末から x-controller1.cc.u-tokyo.ac.jp（X 端末サーバー）にログインして、ウィンドウのメニューから接続するホストを選択して下さい。なお、X 端末の利用には予め登録が必要です。X 端末の利用及び登録方法についてはスーパーコンピューティングニュース Vol.3 No.3, 2001.5「ユーザーフロア X 端末の使い方」を参照して下さい。

3. 利用者登録

3.1. 大型計算機システムの利用登録

本センターの UNIX システム（バルク以外）を利用するためには

「大型計算機システム利用申請書」

を提出して利用者番号を取得します。ただし、利用者番号を取得した時点では UNIX システムには登録がありませんので、利用するシステムに接続してオンライン登録（newuser 手続き）をする必要があります。詳細は「3.2. スーパーコンピューターシステムの登録」を御覧下さい。

バルク利用システムの場合には

「大型計算機システム（定額負担金バルクコース）利用申請書」

を提出して利用者番号を取得する必要があります。バルク利用システムは取得した利用者番号でログインできます。なお、本利用登録ではバルク利用システム以外のシステムは利用できません。

3.2. スーパーコンピューターシステムの登録

UNIX システム（バルク以外）の利用登録は大型計算機システム利用申請とは別にシステムごとにオンライン登録を行う必要があります。以下に示す newuser 手続きで行います。

(1) mpp-s に接続します。（利用者番号と VOS3 パスワードが必要です。）

```
% telnet  mpp-s.cc.u-tokyo.ac.jp

HI-UX/MPP (mpp-s.cc.u-tokyo.ac.jp) (pts/5)
login: newuser

           System Registration Command

Enter Your (VOS3) Login name : a30000   （利用者番号を小文字で）
Enter Your (VOS3) Password : _______   （VOS3 パスワードを小文字で）
```

☞ 上記パスワードは VOS3 のパスワード又は利用承認通知書に書かれているパスワードを使用します。パスワードが不明の場合には共同利用掛（03-5841-2717）まで御連絡下さい。

(2) ホスト名の一覧が表示されますので mpp-s.cc.u-tokyo.ac.jp を選択します。

```
 1   * m-unix.cc.u-tokyo.ac.jp
 2     mpp-s.cc.u-tokyo.ac.jp
 3     x-controller1.cc.u-tokyo.ac.jp
 4     io-controller.cc.u-tokyo.ac.jp
 5     mathematica.cc.u-tokyo.ac.jp
 6     macsyma.cc.u-tokyo.ac.jp
```

☞ メニュー項目には mpp-s.cc.u-tokyo.ac.jp のみ表示されますが、これを選択すると mpp-p.cc.u-tokyo.ac.jp、mpp-bt.cc.u-tokyo.ac.jp（バッチシステム）にも同時に登録されます。又、ホスト名の先頭に * 印が付いているものは既に登録されているホストです。

(3) 利用登録、取り消し、支払コード（課金番号）変更、ファイル使用量増減の手続きをメニューから選択します。

```
1.     User Account Registration
2.     User Account Deletion
3.     Account Code Change
4.     Increase File Size Limit
5.     Decrease File Size Limit
```

☞ newuser 手続きで利用取消、支払コード変更、ファイル使用量増減が可能です。（利用登録以外を選択した場合、mpp-s.cc.u-tokyo.ac.jp のパスワードの入力が必要となります。）

(4) 利用登録時には /home ファイル使用量の上限値の設定が必要です。メッセージに従って間違いの無いように設定して下さい。

```
Specify your /home size(MB)[default 200MB, max 8192MB]:
```

☞ ファイル上限値と課金額が表示されますので確認して下さい。確認メッセージに応答した時点でファイル課金が発生しますので慎重に入力して下さい。なお、/short ファイル使用量はこの手続きとは別に実際に保存しているファイル量に従って課金します。

なお、UNIX システムへの登録関連の反映の契機は以下の様になっています。

- 登録手続きが完了すると即時登録（5 分程かかることがあります）されます。
- 登録の抹消は取り消し手続きを行った月の月末となります。
- ファイル使用量上限値の増加は即時変更、削減は翌月から変更となります。
- 支払コードの変更は即時反映されます。

newuser 手続きによる登録を行った場合の注意事項です。

- 本手続きによる取り消し処理を利用者自身が行わない限り UNIX システムの登録は翌月に継続します。UNIX システムの取り消しを希望する場合には、必ず取り消し処理を行って下さい。
- 大型計算機システム利用登録を継続している場合には年度を越える場合でも UNIX システムは継続扱いとなります。（前年度使用していた支払コードが取り消された場合は別の支払コードが割り当てられます。）
- 利用負担金見込み額を超えた場合でも本手続きによる取り消し処理をしない限り、UNIX システムの登録は継続します。（利用負担金見込み額を超えた分についても課金されます。）
- 大型計算機システム利用登録上の変更（基本負担金コースの変更、支払コードの削除等）があった場合には翌月の課金額、支払コードが変更となる場合があります。（前月使用していた支払コードが取り消された場合は別の支払コードが割り当てられます。）

次の手続きは「newuser」ではできません。「大型計算機システム届出書」により手続きして下さい。

- センター利用（大型計算機システム）の取り消し
- 支払コードの削除
- 基本負担金額（月額 1,000 円/2,000 円）の変更
- VOS3 システムのファイル使用量上限値の変更
- 利用負担金見込み額の変更

3.3. 運用支援システムおよび各種サーバーの登録

運用支援システムおよび各種サーバーへの利用登録は以下に示す newuser 手続きで行います。

(1) m-unix に接続します。（利用者番号と VOS3 パスワードが必要です。）

```
% telnet  m-unix.cc.u-tokyo.ac.jp

HI-UX/MPP (m-unix.cc.u-tokyo.ac.jp) (pts/5)
login: newuser

            System Registration Command

Enter Your (VOS3) Login name : a30000   （利用者番号を小文字で）
Enter Your (VOS3) Password : _______  （VOS3 パスワードを小文字で）
```

☞ 上記パスワードは VOS3 パスワード又は利用承認通知書に書かれているパスワードを使用します。パスワードが不明の場合には共同利用掛（03-5841-2717）まで御連絡下さい。

(2) ホスト名の一覧が表示されますので登録するホストを選択します。

```
1     m-unix.cc.u-tokyo.ac.jp
2     mpp-s.cc.u-tokyo.ac.jp
3     x-controller1.cc.u-tokyo.ac.jp
4     io-controller.cc.u-tokyo.ac.jp
5     mathematica.cc.u-tokyo.ac.jp
7     macsyma.cc.u-tokyo.ac.jp
```

m-unix を選択した場合は(3)、(4)の選択、設定が必要です。

(3) 利用登録、取り消し、支払コードの変更から選択します。

```
1.     User Account Registration
2.     User Account Deletion
3.     Account Code Change
```

(4) 支払コードを設定すると手続きは終了です。

```
Specify account code[default: A](or quit):
```

☞ ここで指定した支払コードはプリンター出力の課金に使用されます。

4. ログインとパスワード

4.1 ログイン

スーパーコンピューターシステムのインタラクティブ（TSS）処理用サブシステム及び運用支援システムは一般の UNIX システムと同様にログイン名とパスワードを入力することでログインして利用できます。なお、利用者番号をログイン名として使用するときは英文字（大文字）を小文字で指定して下さい。

```
% telnet  mpp-s.cc.u-tokyo.ac.jp
Trying...
Connected to mpp-s.cc.u-tokyo.ac.jp.
Escape character is '^]'.

HI-UX/MPP (mpp-s.cc.u-tokyo.ac.jp) (pts/18)
login: a30000
```

☞ 初期パスワードは newuser で使用したパスワード（バルク利用システムの場合は代表者宛てに発送した初期パスワード）です。最初のログインでは、メッセージに従って初期パスワードを変更しなければなりません。このとき Old password: には初期パスワードを入力して下さい。なお、パスワードは mpp-s と mpp-p（または bulk-s と bulk-p） とで別々のパスワード管理になります。

```
SR8000/MPP Unix at Computer Centre, University of Tokyo (CCUT).
HI-UX/MPP(Mon  Apr 02 09:30:00 JST  2001)/ mpp-s.cc.u-tokyo.ac.jp.
Information ----
以下ログインメッセージ
```

ログイン名、パスワードが正しいとログインメッセージが表示されます。このメッセージはそのシステムに関する重要な内容が記述されますので必ず目を通すようにして下さい。以下のメッセージはシステム停止予定時刻と起動予定時刻です。定期保守や障害時などシステムを停止する必要がある場合に表示されます。

```
System will shutdown at Sat Apr 7  6:00, 2001
System will start    at Sat Apr 7 11:00, 2001
```

ログインメッセージ後、プロンプト% が出ればログイン完了です。ところで、パスワードの入力を 5 回連続して間違えるとアカウントがロックされます。以後正しいパスワードを入力しても以下のメッセージが出力されログインできなくなりますので御注意下さい。

```
Account is locked -- see Account Administrator.
```

また、利用負担金額が見込み額を超えた場合でもログインできません。

```
Login incorrect (you do not have budget)
```

これらの場合には共同利用掛（03-5841-2717）までご連絡ください。

☞ mpp-p.cc.u-tokyo.ac.jp または bulk-p.cc.u-tokyo.ac.jp は同じログイン名で同時に 2 つ以上ログインすることはできません。（Too many allocate node）又、ログインノードが不足している場合にもログインできません。（Can’t allocate a node. Try again later）

ログインが完了したら、センターで用意しているコマンド show-info を実行してみて下さい。このコマンドはセンターからのお知らせを端末に表示するもので、サービス休止や障害に関する情報、FAQ など「センターからのお知らせ」を日本語で掲載しています。重要な情報もありますので必ず目を通すようにして下さい。

```
% show-info
  0 Announce
  1 General information (2001.04.02)
  2 Frequently Asked Questions (2001.04.02)
  a Articles of the Supercomputing news
  q (quit)
  number/character:
```

番号を選択すると、内容が表示されます。画面が停止したら、スペースキーで続きを読むか、「q」キーでメニューに戻ります。

4.2. ログアウト

ログアウトする場合には logout または exit コマンドで終了します。

☞ このとき不要なプロセス（実行中のプログラム）が残っていないか確認し、残っている場合は kill して下さい。

```
% ps -e
UID    PID    TTY     S       TIME CMD
12026 369661 pts/1    S  +    0:00.53   -csh
12026 375324 pts/3    S       0:00.09   a.out
0     375370 pts/3    R  +    0:00.34   ps
% kill -9 375324  （a.out が不要なプロセスの場合）
```

4.3. パスワードの変更

パスワードはシステムのセキュリティ向上のために定期的に変更して下さい。変更方法は以下の通りです。

```
% passwd
Changing password for a30000.
Old password:  現在のパスワード
Last successful password change for a30000: Tue Apr 17 15:44:17 2001
Last unsuccessful password change for a30000: NEVER

New password:  新しいパスワード
Re-enter new password:  新しいパスワード（確認）
```

☞ パスワードを 3 ヶ月以上変更していないとログイン時にパスワードを変更するよう要求されます。その際、旧パスワードと新パスワードを入力することでログインできます。そのまま変更せずに 1 年以上経過した場合にはアカウントがロックされログインできなくなります。

パスワード設定の際には以下に注意して下さい。

- 8 文字（またはそれ以上）で英大文字、英小文字、数字、記号を適当に混在させる。
- 平易な語を使用しない。（辞書にある英単語等も避ける）
- 氏名、生年月日、電話番号、利用者番号は使用しない。
- 回文（前後いずれから読んでも同じ）や単語の逆並びを使用しない。

なお、英小文字のみのパスワードや一度使用したものなどシステムが認めないパスワードは設定できません。メッセージに従って別のパスワードを設定して下さい。

5. シェルとコマンド

ログインするとログインメッセージのあとにプロンプト% が現れますが、これは C シェル（csh）と呼ばれるコマンドインタープリター（解析プログラム）が入力要求のメッセージとして出力しています。つまり、UNIX のコマンドを入力できる状態です。文字列を入力するとシェルがオプションや引数を展開してコマンドに実行を移します。

```
%   ←プロンプト
% ls  ←コマンド
a.f     a.out     ←実行結果（ファイルがないときは表示されません）
%
```

コマンドが実行を終えると、プロンプト% が表示され、制御はシェルに戻ります。

☞ シェルには csh 以外に sh（Bourne シェル）、ksh（Korn シェル）等がありますが、ここでは本センター標準の csh を使用しているものとして記述します。（tcsh, bash はサポートしていません。）

C シェルやコマンドの使用方法はそのほとんどが標準的な UNIX システムのコマンドと同じなので一般の書籍（UNIX コマンドの解説書など）が十分参考になります。ただし、機種や OS によってオプションや機能には若干の違いがあるため、必要に応じてオンラインマニュアル（man コマンド）でコマンドのオプションや機能を確認して下さい。

```
%  man ls
```

☞ 端末に合わせて次のように設定することで日本語で表示することもできます。

```
% setenv LANG ja_JP.SJIS  （あるいは ja_JP.eucJP）
```

英語に戻すときは以下のようにします。

```
% unsetenv LANG
```

内容が表示され、画面が停止したら、スペースキーで続きを読むか、「q」キーでシェルに戻ります。使い方のわからないコマンドについてもこのように調べます。

次にファイル、ディレクトリー操作に関する主なコマンドの例を挙げます。

コマンド	機能	例
ls	ファイル情報の表示	% ls –l
cat	ファイル内容の表示、結合	% cat a.f b.f
more	ファイル内容の表示	% more a.f

less	ファイル内容の表示	% less a.f
vi	ファイルの作成、編集	% vi a.f
grep	パターン検索	% grep abc a.f
cp	ファイルのコピー	% cp a.f b.f
rm	ファイルの削除	% rm –i a.f
mv	ファイルの移動（ファイル名変更）	% mv a.f b.f
cd	ディレクトリーの移動	% cd work
pwd	カレントディレクトリーの表示	% pwd
mkdir	ディレクトリーの作成	% mkdir work
rmdir	ディレクトリーの削除	% rmdir work

例えば ls コマンドはファイルの情報（パーミッション、所有者、グループ、作成日付など）を見ることができます。

```
% ls –l a.f
-rw-------   1 a30000   users              43 Jan 31   2001 a.f
```

プログラムの作成や編集を行う場合にはシステム標準のエディターである vi を使用します。作成、編集しようとするファイル名が a.f のとき以下のように起動します。

```
% vi   a.f
```

vi エディターの使用方法は本誌別記事「vi エディターの使い方」を御覧下さい。プログラムが完成したら、ファイルに保存して、エディターを終了します。

作成したファイルの内容を端末の画面に表示するには cat コマンドがあります。

```
% cat   a.f
c  test
      write(*,*) ‘abc’
      end
```

ただし、このコマンドはファイルの行数が端末の画面より行数が多いと流れていってしまいます。このため、画面が一杯になったら、一度停止してスペースキーの入力を待つコマンドがあります。

```
% more  a.f
```

さらにファイルの内容に漢字が含まれている場合には次のコマンドを使用して下さい。

```
% less  a.f
```

☞ more 及び less は途中でプロンプトに戻る（“ q ”）、１行前に戻す（“ k ”）、１行先に進める（“ j ”）という操作もできます。

以下 UNIX コマンドや C シェルを使用する上で知っておきたい機能、利用において便利なコマンドについて説明します。

標準入力と標準出力

UNIX のコマンドやプログラムは通常、データーの入力（標準入力）が「端末のキーボード」、実行結果の出力（標準出力）が「端末の画面」となっていますが、コマンドやプログラムの実行結果を「端末の画面」ではなく、「ファイル」に出力することができます。

```
% cat  a.f  >  b.f
```

画面には表示されず、ファイル b.f が作成されて a.f の内容が書き込まれています。これを、標準出力を変更（リダイレクト）する、と言います。また、

```
% cat  a.f  >>  b.f
```

としてファイルに追加書きすることもできます。追加書きではなく、上書きするには

```
% cat  a.f  >! b.f
```

とします。

☞ 出力をリダイレクトしてもエラーメッセージだけは画面に表示されます。エラーメッセージは特別な出力先（標準エラー出力）なので、必要な場合は以下のようにリダイレクトします。

```
% cat  c.f  >&  e.f
```

また、標準入力もリダイレクトできます。以下の grep コマンドは「端末のキーボード」からではなく、「ファイル」からデーターを入力します。

```
% grep  abc  <  a.f
```

この例ではファイル a.f 中の文字列 abc を含む行を表示します。

☞ 入力が必要なコマンドはリダイレクトしないと入力待ちとなり、プロンプトに戻りません。入力の終わりを指示する場合には Ctrl-D（Ctrl キーを押しながら D キーを押す）を使用して下さい。

```
% grep  abc
```

なお、パイプという機能を使っても同様のことができます。

```
%  cat a.f | grep abc
```

パイプ（|）は標準出力の内容を次のコマンドに渡す役割をします。 a.f の内容は cat の標準出力からパイプを通して grep の標準入力に渡され、文字列を検索した結果は grep の標準出力（画面）に表示します。

これらの入出力リダイレクション機能は以下のようにプログラムへのデーター入力や計算結果のファイルへの出力に大変便利ですので憶えておくと良いでしょう。

```
%  program < data > result
```

初期設定ファイル

C シェルはログイン時に自動的に各利用者のホームディレクトリーにある以下の初期設定ファイルを読み込み、環境設定を行います。

```
.cshrc      .login
```

これらのファイルはコマンド検索パス（コマンドの置いてあるディレクトリー名のリスト）等のシェルの環境設定に使用されます。これらのファイルは変更も可能ですが、トラブル

の原因となりますので特に必要がない限り変更しないで下さい。なお、センター作成の初期ファイルに戻す場合には以下のファイルをホームディレクトリーにコピーしてください。

```
/usr/local/skel/.cshrc
/usr/local/skel/.login
```

☞ ピリオドで始まるファイルを ls で見るには以下のようにします。

```
% ls –a
```

プロセスの強制終了

実行中のプログラムやコマンドのことをプロセスといいます。時間のかかるプログラムを実行したが一度止めたい、コマンドが入力待ち状態となって先に進まないなどのようにプロセスを途中で停止したい場合には、

Ctrl-C （Ctrl キーを押しながら C キーを押す）

を使用します。コマンド、プログラムが終了するとシェルのプロンプト% に戻ります。これでも停止しないときは

Ctrl-Z （Ctrl キーを押しながら Z キーを押す）

でプログラムを一時停止し、次のようにプロセスを強制終了します。

```
% ps -e
UID   PID    TTY     S       TIME CMD
12026 369661 pts/1   S  +    0:00.53   -csh
12026 375324 pts/3   S       0:00.09   a.out
0     375370 pts/3   R  +    0:00.34   ps
% kill -9 375324  （a.out が実行中のプログラムの場合）
```

Ctrl-Z による停止も効かないときは､別の端末からログインして上記操作 kill を行います。

ファイル転送

ファイル転送には ftp コマンドを使用します。ログイン名とパスワードの問い合わせに答えてホストに接続し、以下のサブコマンドを使用してファイルを転送します。

サブコマンド	機能	例
put	ファイル送信	ftp> put local remote
get	ファイル受信	ftp> get remote local
binary	バイナリーデーターの転送	ftp> binary
ascii	テキストファイルの転送	ftp> ascii
ls	ファイル一覧表示（リモート）	ftp> ls
cd	ディレクトリーの移動（リモート）	ftp> cd remote_dir
bye	ftp コマンドの終了	ftp> bye

```
% ftp m-unix.cc.u-tokyo.ac.jp
Connected to m-unix.cc.u-tokyo.ac.jp.
220 m-unix.cc.u-tokyo.ac.jp FTP server (HI-OSF/1-MJ) ready.
Name (m-unix.cc.u-tokyo.ac.jp:a30000): a30000  ←ログイン名
331 Password required for a30000.
Password:  ←パスワード
230 User a30000 logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

電子メール

電子メールの送信及び受信を行うことができます。メールアドレスはログイン名@ホスト名.cc.u-tokyo.ac.jp となります。ログインするホストによって異なりますので注意して下さい。

送信

```
% mail a30000@mpp-s.cc.u-tokyo.ac.jp（送り先のメールアドレス）
Subject: test  ←サブジェクト
test mail  ←本文
.      ←.（ピリオド）のみの行
Cc:      ←カーボンコピー（不要なら空欄）
%
```

標準入力を利用してファイル testmail の内容を送ることもできます。

```
% mail a30000@mpp-s.cc.u-tokyo.ac.jp < testmail
```

受信

```
% mail
mailx [5.2 UCB] [OSF/1]  Type ? for help.
"/var/spool/mail/a30000": 1 message 1 new
>N  1 a30000           Wed Aug 15 18:48  12/384  "test"
? 1   ←メール番号
 --ヘッダーと本文が表示されます--
? q  ←コマンドの終了
Saved 1 message in mbox
%
 --メールは mbox というファイルに保存され、メールスプールから削除されます--
```

なお、mbox に保存されたメールを読むときは以下のようにします。

```
% mail –f mbox
```

サブコマンド	機能	例
番号	メールの表示	? 2
h	メール一覧表示	? h
s	ファイルへ保存	? s file
d	メールの削除	? d 1
x	コマンドの終了（何もせず終了）	? x
q	コマンドの終了（mbox へ保存）	? q

☞ 本システムは POP3、IMAP4 などをサポートしておらずメールクライアントソフトウェア(Eudora、Outlook など）は使用できません。

プリンター印刷

センターユーザーフロア設置のプリンターを使用することができます。例えばテキストファイルの印刷は以下のように print コマンドを使用します。

```
% print –text file
```

その他、両面印刷やポストスクリプト印刷、超高画質カラープリンターの利用等ができます。詳細はスーパーコンピューティングニュース Vol.3 No.3, 2001.5「UNIX システム用プリンター装置の使い方」を御覧下さい。

☞ プリンターを使用する場合は m-unix に登録が必要です。（バルク利用システムからプリンターの使用はできません。）

課金情報

利用見込み額や課金額など課金に関する情報は la コマンドで出力できます。（バルク利用システムは表示内容が異なります。）

```
% la
budget data for a30000 at Thu Aug 16 04:31:04 2001
 acct    course    budget     fare   balance  cpu_used(HH:MM:SS)
  *A       2000    999000    78791    920209     159:51:51
```

支払コード　円コース　利用見込額　課金額　残額　使用 CPU 時間（当該月）

複数の支払コードを持つときは -a オプションを指定します。*印のあるものが標準支払コード（デフォルトの支払コード）です。

```
% la –a
budget data for a30000 at Thu Aug 16 04:31:04 2001
 acct    course    budget     fare   balance  cpu_used(HH:MM:SS)
  *A       2000    999000    78791    920209     159:51:51
   C       2000   1000000    10711    989289       0:00:00
```

☞ 課金額には基本負担金、CPU 課金、ファイル課金、プリンター課金等全てが含まれています。課金情報の更新は通常 1 日 1 回です。

ディスクに関する情報については –d オプションを使用します。

```
% la -d
disk resource for a30000
  file system     occupied(MB)  limit(MB)
  /home                  235       4096
  /short/mpp-s             1         --
  /short/mpp-p             1         --
  /short/mpp-bt            1         --
```

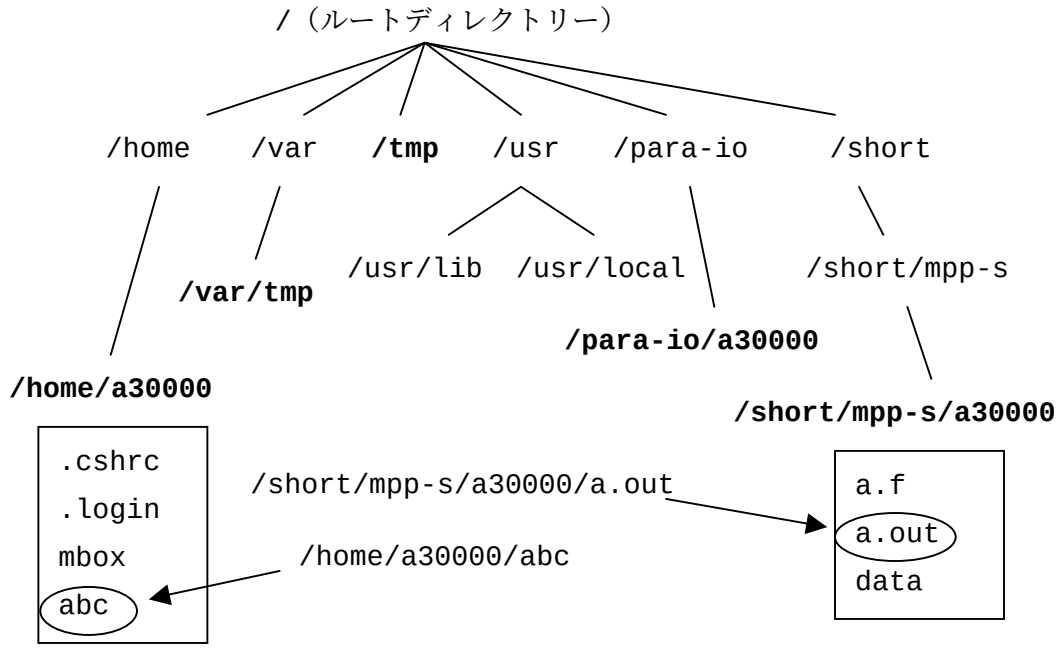
ディレクトリー名　ファイル使用量　ファイル使用量上限値

6. ファイルとディレクトリー

UNIX システムにおいてはプログラムやデーター、文書などの情報は「ファイル」という単位でディスク上に格納されています。格納する場所は「ディレクトリー」と呼ばれるツリー構造によってグループ分けされており、利用者ごとに区画が割り当てられています。以下の例のようにファイルの保存されている場所をディレクトリーによって指し示すことができます。

/	ルートディレクトリー
/home	長期保存ファイルディレクトリー
/home/a30000	利用者 a30000 のホームディレクトリー
/home/a30000/work	利用者の作成した work ディレクトリー
/home/a30000/work/program.f	利用者のファイル program.f

本センターの UNIX システムは以下の図のようなディレクトリー構成としています。ただし、ファイルを作成できるのは利用者用に開放されたディレクトリー（例えば図中の太字部分）です。



利用者が使用できるディレクトリーとして以下を用意しています。

```
/home/{ログイン名}
/short/{ホスト名}/{ログイン名}
/para-io/{ログイン名}     （m-unix は除く）
/tmp
/var/tmp
```

/home

ホームディレクトリーと呼ばれる利用者が主に使用するディレクトリーです。ここに作られたファイルは通常、利用者自身で削除するまで保存されます。

/short

利用者が作業用に使用するディレクトリーです。ここに作られたファイルは作成（または最終更新）から 15 日後に削除されます。

/para-io（m-unix は除く）

バッチジョブで使用するデーターを高速に並列入出力を行うディレクトリーです。大容量の単一ファイルの入出力に有効でバッチジョブからのみアクセスできます。ここに作られたファイルは作成（または最終更新）から 5 日後に削除されます。

/tmp

システムおよび利用者が短時間の作業用に使用するディレクトリーです。ここに作られたファイルは作成終了後しばらくして（平均 1 時間程度。最低 30 分、最大 1 時間 30 分以内で）削除されます。

/var/tmp

一時ファイルの作成場所として主にエディターや Gaussian98（m-unix を除く）が使用します。ここに作られたファイルは作業（ジョブ）終了後に削除されます。

各システムのディレクトリーごとの設定及びファイル使用量上限値については「8.3. ディスク資源」を参照して下さい。

7. コンパイルと実行

7.1. FORTRAN

FORTRAN プログラムのコンパイル、リンクは f77 または f90 コマンドを使用します。

```
f77  または f90
    ファイル名..
    [-c]
    [-o ロードモジュールファイル名]
    [-コンパイルオプション..]
    [-リンケージオプション..]
    [-オプション..]
    [-I インクルードディレクトリー名]
    [-L ライブラリーディレクトリー名]
    [-l ライブラリー名]
    [-i,言語仕様拡張オプション]
```

使用例

```
% f77  -parallel -o loadmodule  main.f  sub.f  -lmatmpp
```

FORTRAN プログラムのソースファイル名は通常 .f で終わる名前を使用します。大文字で.F とした場合は C 言語プリプロセッサーを通した後、FORTRAN のコンパイルを行います。コンパイルが正常に終了するとリンクが行われ、ロードモジュール a.out が作成されます。-o オプションを使用すれば生成するロードモジュール名を指定することができます。なお、オプションは指定した順番（左から右）に処理されます。ファイル名の前後に置くことができますが、ライブラリー名を指定する-l オプションはライブラリーを呼び出すプログラムより後に指定して下さい。

```
% f77 –c -parallel  main.f  sub.f
% f77 –parallel –o loadmodule main.o sub.o –lmatmpp
```

-c オプションを指定した場合はオブジェクトモジュール.o を生成します。オブジェクトモジュールをファイル名として指定するとリンクを行ない、ロードモジュールを生成します。MPI 機能を使用したプログラムをコンパイルする場合には以下のコマンドを使用します。

```
mpif77  または mpif90
    ファイル名..
    [-c]
    [-o ロードモジュールファイル名]
    [-オプション]
```

```
% mpif77  sample.f  -o program
```

☞ MPI 機能のコマンドを使用する場合には以下に示す環境変数 path の設定が必要です。

```
% set path=($path /usr/mpi/bin)
```

これらのコマンドは f77 または f90 コンパイラーで MPI 機能を使用する場合に必要なヘッダーファイルやリンクするライブラリー、コンパイルオプションを自動的に設定します。-オプションを指定してコンパイラーにオプションを渡すこともできます。

PARALLELWARE（Express）機能を使用しているプログラムをコンパイルする場合は以下のコマンドを使用します。

```
srf77  または srf90
    ファイル名..
    [-オプション]
```

```
% srf77  -kXH  host.f  -o host
% srf77  -kXN  node.f  -o node
```

HPF 言語仕様で書かれているプログラムは Parallel FORTRAN トランスレーターを使用して FORTRAN90 ソースプログラムに変換します。

```
pf90
    ファイル名..
    [-オプション]
```

```
% pf90  sample.f
```

変換したソースプログラムは f90 コマンドでコンパイルします。その際、HPF ライブラリーをリンクする必要があります。

```
% f90  pf_node/sample.f  -lhpf
```

7.2. C、C++

C プログラムのコンパイルは cc、C++プログラムは CC コマンドを使用します。

```
cc  または  CC
    [オプション]
    ファイル名..
```

☞ C は ISO/IEC9899 及び ANSI X3.159-1989 規格に準拠（-noansi オプションで K&R 互換仕様にも対応）、C++は The Annotated C++ Reference Manual に準拠しています。

使用例

```
% cc  -parallel -o loadmodule  main.c  sub.c  -lm
```

C プログラムのソースファイル名は通常 .c で終わる名前を使用します。C++プログラムの場合は.C です。コンパイルが正常に終了するとリンクが行われ、ロードモジュール a.out が作成されます。-o オプションを使用すれば生成するロードモジュール名を指定することができます。なお、ライブラリー名を指定する-l オプションはライブラリーを呼び出すプログラムより後に指定して下さい。

```
% cc  -c -parallel  main.c  sub.c
% cc  -parallel -o loadmodule  main.o  sub.o  -lm
```

-c オプションを指定した場合はオブジェクトモジュール.o を生成します。オブジェクトモジュールをファイル名として指定するとリンクを行ない、ロードモジュールを生成します。

MPI 機能を使用した C プログラムをコンパイルする場合には以下のコマンドを使用します。

```
mpicc
    ファイル名..
    [-c]
    [-o ロードモジュールファイル名]
    [-オプション]
```

☞ MPI 機能のコマンドを使用する場合には以下に示す環境変数 path の設定が必要です。

```
% set path=($path /usr/mpi/bin)
```

このコマンドは cc コンパイラーで MPI 機能を使用する場合に必要なヘッダーファイルやリンクするライブラリー、コンパイルオプションを自動的に設定します。-オプションを指定してコンパイラーにオプションを渡すこともできます。

PARALLELWARE（Express）機能を使用しているプログラムをコンパイルする場合は以下のコマンドを使用します。

```
srcc
    ファイル名..
    [-オプション]
```

7.3. 実行

ロードモジュールは実行ファイルとも呼ばれ、ファイルに実行権があるため、ロードモジュール名をコマンドのように入力（バッチジョブの場合はスクリプトファイルに記述）することでプログラムを実行することができます。なお、FORTRAN プログラムの場合は実行時オプションを与えることができます。

```
a.out  またはロードモジュール名
    [-F,'実行時オプション,..']
```

使用例

```
% a.out  –F,’port(stduf)’
```

☞ 要素並列化したロードモジュールはスカラー処理用のサブシステム mpp-s、bulk-s では以下のエラーメッセージを出力し、実行できません。

```
a.out: element parallel ANY: can't execute on jobtype(SS)
```

また、複数ノードを使用した並列実行もできません。mpp-p、bulk-p で実行（2 ノードまで）するか、バッチジョブで並列処理用のジョブクラスを使用します。

コンパイルしたプログラムは通常、上記のような a.out 実行が可能ですが、並列アプリケーションによっては別途起動コマンドが必要な場合もあります。例えば MPI 機能を使したプログラムを実行する場合には以下のコマンドを使用します。

```
mpirun
    [-n ノード数]
    [-np プロセス数]
    [-オプション]
    ロードモジュール名
```

または

```
mpiexec
    [-N ノード数]
    [-n プロセス数]
    [-オプション]
    ロードモジュール名
```

使用例

```
% mpirun  -n 2 –np 8 a.out
```

☞ MPI のコマンドを使用する場合には以下に示す環境変数 path の設定が必要です。

```
% set path=($path /usr/mpi/bin)
```

各ノードにプログラムを配置して並列実行するコマンドが prun です。指定した数のノードでプログラムが実行します。ホストプログラムを作成しない、あるいはアプリケーション専用の実行コマンドを使用しない並列実行にはこのコマンドを使用します。

```
prun
    [-n ノード数]
    [-オプション]
    ロードモジュール名

% prun  -n 2  a.out
```

または

```
prun
    [-f 定義ファイル名]
    [-オプション]

% prun -f  sample.def
```

```
定義ファイル
# sample.def          [# コメント]
ALL:                  [パーティション名:]
*2 a.out              [*ノード数] [プログラム名]
```

本センターでは並列プログラム実行時のパーティション名をデフォルトで設定していますので通常はパーティション名の指定の必要はありません。パーティション名が必要な場合には「ALL」を指定します。

```
% mpirun  -p ALL -n 2 –np 8 a.out
% prun  -p ALL -n 2  a.out
```

なお、デフォルトのパーティション名は環境変数 DEFPART に設定されています。

なお、バッチジョブによる実行方法は上記コマンドをスクリプトファイルに記述してバッチシステムに投入します。バッチジョブの使用方法については「9. NQS（バッチジョブ）」を御覧下さい。

8. システム資源

本センターの計算機は多くの利用者が共通に使用するため、資源占有や性能低下を防ぐ目的で各種システム資源に上限値を設けています。ここでは利用者が SR8000 システムを使用する上でのシステム資源設定について述べます。

8.1. CPU 資源

インタラクティブ環境（TSS）ではログイン時間を制限しています。ログインから 18 時間、無入力（キー入力がないとき）の場合は 2 時間で自動的にログアウトするように設定しています。さらにプロセスごとに CPU 資源制限をしています。この制限値は limit コマンドで確認することができ、先述の自動ログアウト時間以内ならば拡大できます。

```
% limit
cputime         1:00:00             CPU 使用時間制限
% limit  cputime  unlimited
```

バッチジョブ環境の CPU 資源制限はスクリプトファイル中のオプションで設定します。経過時間と CPU 時間による制限がありますが、通常は経過時間により設定します。設定方法の詳細は「8. NQS（バッチジョブ）」を御覧下さい。

```
#@$-lT          リクエスト（ジョブ）の経過時間制限
#@$-lt          プロセスごとの CPU 時間制限
#@$-lE          #@$-lT と同じ
```

最大値はジョブクラス（キュー）ごとに異なります。ジョブクラス制限値（スーパーコンピューティングニュースの表紙裏）を御覧下さい。

8.2. メモリー資源

メモリー資源はジョブリクエスト、あるいはプロセス（実行中のプログラム）ごとに制限します。このため、ジョブクラス及びログインセッションに 1 ノード当たりのメモリー使用量の制限値（ジョブクラス制限値を参照）を設定して資源管理を行っています。説明のために UNIX システムのプロセスごとのメモリー構成を以下に示します。

テキスト領域（.text）

プログラム（実行命令等）を配置します。

データー領域

- 初期化データー（.data）

  初期値のある静的変数を配置します。

- 未初期化データー（.bss）

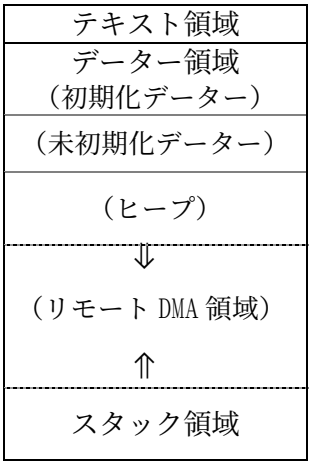
  初期値なしの静的変数を配置します。

- ヒープ

  プログラム実行時に動的に配置します。

- リモート DMA 領域

  ノード間転送を行う場合に配置します。



スタック領域

ローカル変数、関数呼び出し時の引数、戻り値等を配置

☞ プログラムの大きさ（.text+.data+.bss 他）は size コマンドで確認できます。

```
% size –f  a.out
```

このうち利用者が上限値を設定できるのは使用するメモリーの大きさ（仮想メモリーサイズ）とデーター領域及びスタック領域です。例えば FORTRAN プログラムでは配列などの変数は通常データー領域に配置しますので不足する場合は仮想メモリーサイズとデーター領域を拡張します。また、C プログラムでは自動変数（auto）がスタック領域を使用しますのでスタックが不足することがあります。この場合はスタック領域を拡張するか、配列を静的変数（static）で確保し、データー領域を使用して下さい。

インタラクティブ環境（TSS）のメモリー資源制限は limit コマンドで確認することができ、システムで設定した上限値以内なら変更も可能です。

```
% limit
datasize        131072 kbytes      データーサイズ
stacksize       1024 kbytes        スタックサイズ
memoryuse       unlimited          実メモリーサイズ
64datasize      131072 kbytes      データーサイズ（64 ビットモード）
64stacksize     1024 kbytes        スタックサイズ（64 ビットモード）
addressspace    128 Mbytes         仮想メモリーサイズ
pgrpadrspace    128 Mbytes         仮想メモリーサイズ（プロセスグループ）
```

データー領域が不足するときはデーターサイズを拡大又は解除します。但し、制限を解除してもシステムの設定値を超えることはできません。（スタック領域も同様です。）

```
% limit  datasize  unlimited
% limit  stacksize 32M
```

64 ビットモードで作成したプログラム（-64 オプションを指定したコンパイル）の実行は以下の項目を変更します。

```
% limit  64datasize  unlimited
% limit  64stacksize  32M
```

なお、これらの設定値が仮想メモリーサイズを超える場合は以下の項目も変更して下さい。

```
% limit  addressspace  unlimited
% limit  pgrpadrspace  unlimited
```

バッチジョブ環境のメモリー資源制限はスクリプトファイル中のオプションで設定します。

#@$-lM	リクエスト（ジョブ）ごとのメモリーサイズ
#@$-lm	プロセスごとのメモリーサイズ
#@$-ld	プロセスごとのデーターサイズ
#@$-ls	プロセスごとのスタックサイズ

これらのオプションは limit コマンドと異なり、64 ビットモードを区別しません。オプションの設定方法の詳細は「9. NQS（バッチジョブ）」を御覧下さい。

以下はメモリー使用に関する注意事項です。

(1) 使用するメモリーの大きさが 2GB を超えるプログラムはデフォルトの 32 ビットアドレッシングモードでコンパイルしてもロードモジュールを作成できません。この場合は-64 オプション（C では-lp64 オプション）を指定して、64 ビットアドレッシングモードのロードモジュールを作成する必要があります。

(2) 静的自動リモート DMA 機能（コンパイル時に-rdma オプションを指定）を使用している場合、プログラム中の全静的データーがリモート DMA 領域に確保されるため、より多くのメモリーを必要としますので注意して下さい。

(3) 複数のプロセスをノード内に生成（ノード内 MPI など）するときはメモリー不足に注意して下さい。特に静的自動リモート DMA 機能を使用している場合にはノード内にプロセスを重ねることができませんので、機能を外すことで対処して下さい。

(4) 複数ノードのジョブは確保したノード数分のメモリーを使用できます。ただし、プロセスが各ノードに分散して配置されるため、全メモリーを使用する大規模配列を単純に定義することはできず、プログラム上で配列を各ノードに分散する必要があります。

8.3. ディスク資源

ディスク資源制限は利用者ごと（バルクシステムではグループごと）にファイル使用量上限値（quota）で制限しています。上限値を超えてファイルを作成しようとするとエラー

```
Disc quota exceeded
```

となりますので、不要なファイルの削除や上限値の変更で対処して下さい。なお、ファイル使用量及び上限値は以下のコマンドで確認できます。

```
% la  –d
```

また、ディスク資源は全ての利用者で共有しているため、無駄なファイルの保存はディレクトリーを圧迫します。不要になったファイルは残しておかないようお願い致します。また、ファイル数にも限界があります。ファイルの容量が小さくても数が多い場合には一つにまとめる（tar コマンドでアーカイブする）などディスクを効率的に利用するよう心掛けて下さい。なお、ファイル保存にあたっては万一の事故に備えて大切なファイルは各自でバックアップをとっておかれますようお願い致します。

以下は各システムのディレクトリーにおけるファイル使用量の上限値及び保存期間です。

SR8000/MPP　　(2001 年 5 月 7 日現在)

ディレクトリー名	ファイル上限値	保存期間
/home	利用者の宣言値	―
/short/mpp-s	制限なし	15 日間
/short/mpp-p	〃	〃
/short/mpp-bt	〃	〃
/para-io	〃	5 日間

注）/home、/short はどのサブシステムからでもネットワーク（NFS）経由で読み書きできますが、/home はバッチシステム、 /short/{ホスト名} は各ホスト（サブシステム）に直結しています。なお、/para-io はバッチジョブからのみ利用可能です。

/home　newuser 手続きにより利用者ごとにファイル使用量上限値を設定しています。最初の登録時に上限値を宣言し、利用の取消しや変更をしない限り、上限値は翌月に継続します。

/short　上限値は制限していません。ただし、ファイルを作成すると容量と保存期間について課金されます。

SR8000/128（バルクコース）　　(2001 年 5 月 7 日現在)

ディレクトリー名	ファイル上限値	保存期間
/home	グループ制限値	―
/short/bulk-s	グループ制限値	15 日間
/short/bulk-p	〃	〃
/short/bulk-bt	〃	〃
/para-io	制限なし	5 日間

注）/home、/short はどのサブシステムからでもネットワーク（NFS）経由で読み書きできますが、/home はバッチシステム、 /short/{ホスト名} は各ホスト（サブシステム）に直結しています。なお、/para-io はバッチジョブからのみ利用可能です。

/home　利用申請書の値でプロジェクトグループ毎にファイル使用量上限値を設定しています。利用者ごとの制限は必要に応じてグループ管理者が行います。

/short　現在は制限していませんが、利用状況によってはプロジェクトグループ毎、利用者毎に制限します。

運用支援システム（m-unix）　　(2001 年 5 月 7 日現在)

ディレクトリー名	ファイル上限値	保存期間
/home	100MB	―
/short/m-unix	100MB	15 日間

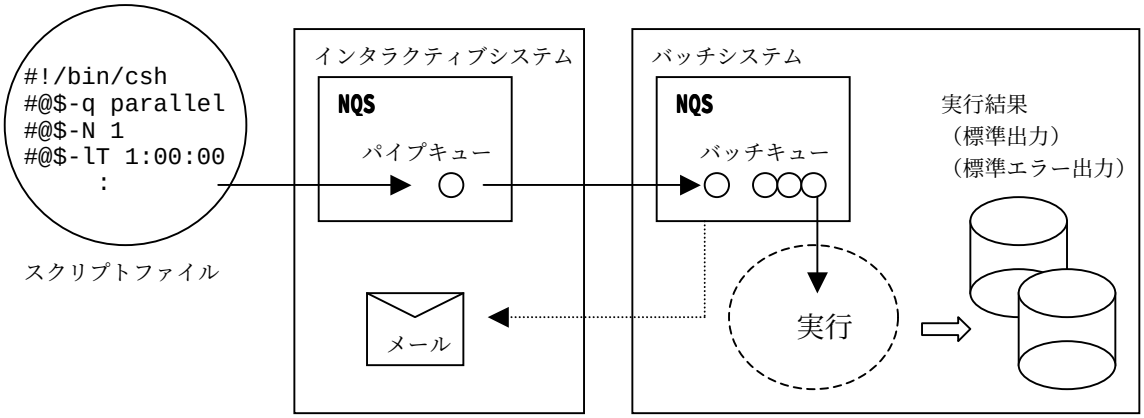
注）ディスク容量が少ないため、ファイル上限値を制限しています。またファイル課金は行っていません。

/home　　100MB に制限しています。（利用状況によっては届出書により 1GB まで拡大できます。）
/short　　100MB に制限しています。（利用状況によっては届出書により 1GB まで拡大できます。）

9. NQS（バッチジョブ）

9.1. NQS とは

NQS（Network Queuing System）は実行形態や資源制限により最適なキューを選択し、ジョブをバッチシステムにキューイングする機能です。キューとはジョブ実行の待ち行列で、利用者が投入したジョブは先に実行しているジョブを待ち、順番がきた時点で実行を開始します。対話形式の TSS とは異なり、実行結果はファイルに残りますのでジョブ投入後はログインしている必要がありません。



9.2. ジョブクラスとキュー

NQS ではジョブを効率よく実行するためにジョブの種類や大きさで分類したジョブクラスを設定しています。ジョブクラスごとに「バッチキュー」と呼ばれるキューが設けてあり、ジョブが使用するノード数やメモリーの大きさ、実行時間制限等の指定によってどのバッチキューに並ぶかが自動的に決められます。また「パイプキュー」と呼ばれるキューがバッチシステムにあるバッチキューへの橋渡しをする役目をします。利用者はパイプキュー名を指定します。本センターのスーパーコンピューターでは以下のジョブクラスを用意しています。

スカラージョブクラス

スカラープログラムを実行するジョブクラスです。ジョブの実行時間（CPU 時間、経過時間）でクラス分けしています。

パイプキュー名　single

バッチキュー名　A、B、C、D、E、F

拡張記憶（ES）使用ジョブクラス

拡張記憶（ES）を使用するジョブ実行のためのジョブクラスです。ジョブの実行時間（CPU 時間、経過時間）でクラス分けしています。

パイプキュー名　single

バッチキュー名　A-ES、B-ES、C-ES、D-ES、E-ES、F-ES

☞　拡張記憶（ES）とはメモリーを外部記憶装置とみなして入出力を行う大容量擬似ファイルシステムです。例えば次のようにファイルとして宣言して利用します。

```
open(60,file='esdata',type='es',space='10GB',form='unformatted')
```

並列ジョブクラス

要素並列プログラムの実行及び複数ノードを使用したジョブの実行を行うジョブクラスです。使用ノード数と実行時間（経過時間）でクラス分けしています。

パイプキュー名　parallel

バッチキュー名　P001、P002、P004、P008、P016

☞　ジョブクラス制限値についてはスーパーコンピューティングニュース表紙裏「SR8000/MPP ジョブクラス制限値」を御覧下さい。上記のほかに以下のジョブクラスがあります。

運用支援システム m-unix のジョブクラス

ジョブの実行時間（CPU 時間）でクラス分けしています。

パイプキュー名　m-unix（省略可）

バッチキュー名　A、B、C、D、E、F、L（L は届出が必要）

バルク利用システム専用キューのジョブクラス

使用可能ノード数は利用申請時の値、ジョブ実行時間は最大 24 時間です。

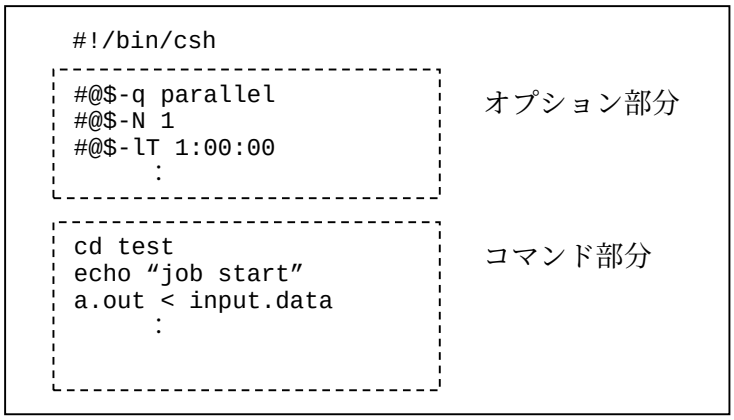
パイプキュー名　プロジェクト名（ログイン名の最初の 3 文字）　例 `#@$-q a00`

バッチキュー名　パイプキュー名と同じ

9.3. スクリプトファイルの作成

バッチジョブを実行するためには、まずジョブを作成しなければなりません。ジョブの作成とは実行するプログラムの手順書を書くようなものであり、その手順書のことをスクリプトファイルと呼びます。スクリプトファイルは UNIX で一般的に使われるシェルスクリプトを使用して記述します。シェルスクリプトにはいくつかの種類（C シェル、Bourne シェルなど）がありますが、ここでは本センターの標準のログインシェルである C シェルスクリプトを使用して説明します。

スクリプトファイルは通常、オプション部分とコマンド部分の 2 つのセクションで構成されます。(1 行目の#!はコマンド部分が C シェルスクリプトで記述されていることを表すものです。)



オプション部分

オプション部分の 1 カラム目は全て#記号(C シェルスクリプトではコメントを意味する)ではじまり、@$に続くオプションによってキューの選択や資源制限を行うことができます。なお、オプションは大文字、小文字を区別しますので意識して記述して下さい。

最初にジョブを投入するキュー(パイプキュー)名を指定します。

```
#@$-q     キュー名
```

ここで指定するキュー名は以下のパイプキュー名です。

```
single
parallel
```

ジョブはこのパイプキューを経由してバッチシステムへ投入されます。次に複数ノードを使用する並列ジョブ(parallel)の場合はノード数を指定します。スカラージョブ(single)の場合、この指定は不要です。

```
#@$-N     ノード数
```

続いてジョブの実行時間(時間:分:秒や分:秒での指定が可能)の見込みを設定します。この時間を過ぎると実行中のジョブでも強制的に終了します。

```
#@$-lT    実行時間制限値(経過時間)
```

☞ 本センターの-lT オプションは経過時間制限値(-lE と同意)としています。標準の-lT(CPU 時間制限値)とは異なりますので御注意下さい。(m-unix は除く)

さらにジョブ全体で使用する 1 ノードあたりのメモリーサイズ制限値を設定します。

```
#@$-lM    メモリーサイズ制限値
```

単位は MB(Megabyte)や GB(Gigabyte)等が使用できます。

拡張記憶（ES）を使用する場合にはパイプキュー single を指定してから、次のオプションを指定します。

```
#@$-lV    拡張記憶（ES）サイズ制限値
```

単位は MB（Megabyte）や GB（Gigabyte）等が使用できます。

以下にオプション部分の記述例を挙げます。

スカラージョブの例

```
#!/bin/csh
#@$-q  single
#@$-lM 2GB
#@$-lT 00:30:00
```

要素並列ジョブの例

```
#!/bin/csh
#@$-q  parallel
#@$-N  1
#@$-lT 01:00:00
#@$-lM 4GB
```

拡張記憶 ES 使用ジョブの例

```
#!/bin/csh
#@$-q  single
#@$-lM 2GB
#@$-lV 10GB
#@$-lT 00:30:00
```

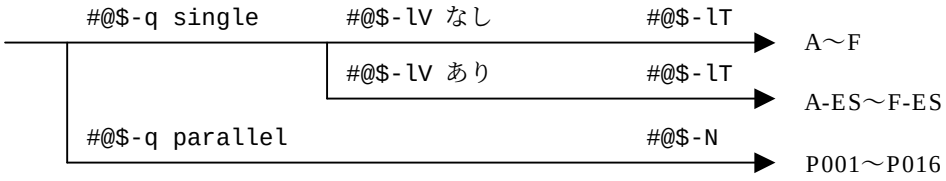
並列ジョブ（複数ノード）の例

```
#!/bin/csh
#@$-q  parallel
#@$-N  4
#@$-lT 01:00:00
#@$-lM 1GB
```

ノード内 MPI ジョブ（複数ノード）の例

```
#!/bin/csh
#@$-q  parallel
#@$-N  2
#@$-lT 01:00:00
#@$-lM 1GB
#@$-J SS
```

バッチキューはこれらのオプションの組み合わせで決定します。図に示すようにパイプキューが single なら拡張記憶（ES）使用の有無とジョブの実行時間でバッチキューが決定します。parallel ならノード数でバッチキューが決定します



主なオプションと指定例の一覧です。

#@$-q	キュー名 （パイプキュー名）	#@$-q parallel
#@$-N	ノード数	#@$-N 4
#@$-lT	実行時間制限	#@$-lT 1:00:00
#@$-lM	メモリーサイズ	#@$-lM 2GB
#@$-lV	拡張記憶（ES）サイズ	#@$-lV 8GB
#@$-AC	支払コード	#@$-AC A

#@$-J	ジョブタイプ	#@$-J SS
#@$-e	標準エラー出力ファイル名	#@$-e file.e
#@$-o	標準出力ファイル名	#@$-o file.o
#@$-lc	コアファイルサイズ	#@$-lc 0MB
#@$-ld	プロセス毎のデーターサイズ	#@$-ld 2GB
#@$-ls	プロセス毎のスタックサイズ	#@$-ls 192MB
#@$-lm	プロセス毎のメモリーサイズ	#@$-lm 2GB
#@$-lt	プロセス毎の CPU 時間制限	#@$-lt 30:00

コマンド部分

コマンド部分にはバッチジョブで実行したいコマンドを並べて書きます。(書き方はシェルスクリプトの記述です。) NQS はこのコマンド部分をホームディレクトリー(/home/ログイン名)で実行します。このため、プログラムやデーターがホームディレクトリー以外にあるときにはディレクトリーを移動するか、パスを指定しなければなりません。例えば work(/home/ログイン名/work)というディレクトリーにある実行ファイル a.out を実行するときには

```
cd work
a.out
```

とスクリプトファイルに記述します。具体的な例は「9.8. 実行例」を参照して下さい。

9.4. ジョブの実行

ジョブの実行はスクリプトファイルをバッチシステムに投入(サブミット)することから始めます。コマンドは qsub を使用します。(以下の例で job.csh は既存のスクリプトファイル名とします。)

```
% qsub job.csh
Request 6128.mpp-s.cc.u-tokyo.ac.jp submitted to queue: parallel
```

NQS はジョブを識別するリクエスト ID(リクエスト番号+ジョブ投入ホスト名)を付加し、パイプキューを経由してバッチシステムにジョブを送り込みます。ジョブの状態(待機中や実行中)は qstat コマンドで知ることができます。

```
% qstat
2001/06/15 (Fri) 15:51:56:   BATCHPIPE REQUESTS on mpp-bt.cc.u-tokyo.ac.jp
NQS schedule stop time : 2001/06/30 (Sat) 08:00:00 (Remain:  352h 13m 52s)
   REQUEST      NAME      OWNER      QUEUE    PRI NICE   CPU   MEM   STATE
6128.mpp-s      job.csh   a30000     A         0   20     960   7168  QUEUED
```

リクエスト ID、スクリプト名、ログイン名が表示されている行が、先程投入したジョブの

状況です。ジョブはバッチキューA で待機（QUEUED）していることがわかります。待機中となる理由には通常の実行待ちのほか、キューに空きがあるのに実行されない次のような場合もあります。①自分のジョブが他のキューで実行（2 本）している②ノードに空きがない③障害、保守等の理由でキューが停止状態となっている等が考えられます。
ジョブが実行を開始すると qstat の結果は実行中（RUNNING）になります。

```
   REQUEST      NAME      OWNER      QUEUE    PRI NICE   CPU   MEM   STATE
6128.mpp-s      STDIN     a30000     A        0   20     960   7168  RUNNING
```

待機中、または実行中のジョブがないときは

```
No requests.
```

となります。また、キュー毎の実行ジョブ、待ちジョブ数も確認できます。

```
%  qstat  -b
2001/06/15 (Fri) 15:54:26:   BATCHPIPE REQUESTS on mpp-bt.cc.u-tokyo.ac.jp
NQS schedule stop time : 2001/06/30 (Sat) 08:00:00 (Remain:  352h 11m 22s)
 QUEUE NAME STATUS    TOTAL   RUNNING  RUNLIMIT  QUEUED  HELD   IN-TRANSIT
 A          AVAILBL   0       0        4         0       0       0
 B          AVAILBL   0       0        4         0       0       0
 C          AVAILBL   0       0        4         0       0       0
 D          AVAILBL   0       0        4         0       0       0
 E          AVAILBL   0       0        3         0       0       0
 F          AVAILBL   0       0        3         0       0       0
 A–ES       AVAILBL   0       0        4         0       0       0
 B–ES       AVAILBL   0       0        4         0       0       0
 C–ES       AVAILBL   0       0        4         0       0       0
  :            :                                 :
```

計画停止時刻について

qstat コマンドの出力には以下のような日付、時刻と残り時間が表示されます。これはセンターがシステム（または NQS）を停止する予定時刻であり、計画停止時刻と呼びます。

```
NQS schedule stop time : 2001/06/30 (Sat) 08:00:00 (Remain:  352h 11m 22s)
```

この計画停止時刻が設定されているとき、この時刻を超える実行時間制限値（#@$-lT）を記述しているジョブは実行されませんので御注意下さい。（実行時間制限値の記述がない場合には各キューで設定している最大の実行時間制限値が指定されているものと仮定します。）

バッチシステムの障害について

バッチシステムに障害が発生した場合、実行中のバッチジョブが異常終了する場合があります。この場合のジョブは障害回復後、キューに自動的に再投入されますので御承知置き下さい。なお、ジョブを再実行したくない場合には予めオプション#@$-nr を設定しておくようにして下さい。

9.5. ジョブの結果

ジョブが終了した場合、オプション（#@$-e、-o、-eo 等）で特に指定していなければ以下の 2 つのファイルが作成されます。プログラムやスクリプトファイルに出力ファイルを作成するように記述している場合にはそのファイルも作成されます。

標準出力：　　スクリプトファイル名.oN
標準エラー出力：　スクリプトファイル名.eN

ここで、N はリクエスト番号で、1〜5 桁、スクリプトファイル名は最初の 7 文字を使用します。 実際には以下のようなファイル名で作成されます。

標準出力：　　job.csh.o6128
標準エラー出力：　job.csh.e6128

通常、標準出力には実行結果が、標準エラー出力にはエラーメッセージが出力されます。ただし、標準出力にエラーが出力されることもありますので両方のファイルを確認する必要があります。

☞ 標準出力、標準エラー出力はプログラムやコマンドを端末で実行したとき、通常なら画面に表示される出力です。また、スクリプトファイルを使用せずに端末からジョブを直接入力（標準入力）した場合には以下のファイルが作成されます。（N はリクエスト番号）
　標準出力：　　STDIN.oN
　標準エラー出力：　STDIN.eN

9.6. ジョブのキャンセル

実行中または待機中のジョブをキャンセルしたい場合には qstat コマンドでリクエスト ID を確認して qdel コマンドでキャンセルします。

```
   REQUEST      NAME      OWNER      QUEUE    PRI NICE   CPU   MEM   STATE
6128.mpp-s      STDIN     a30000     A        0   20     960   7168  RUNNING
```

例えば上のジョブをキャンセルする場合は

```
% qdel –k 6128.mpp-s.cc.u-tokyo.ac.jp
```

ログインしているホストがジョブ投入ホストと同じ（この例では mpp-s にログインしている）ならば

```
% qdel –k 6128
```

のように省略できます。なお、待機中（QUEUED）のジョブの場合は –k は不要です。

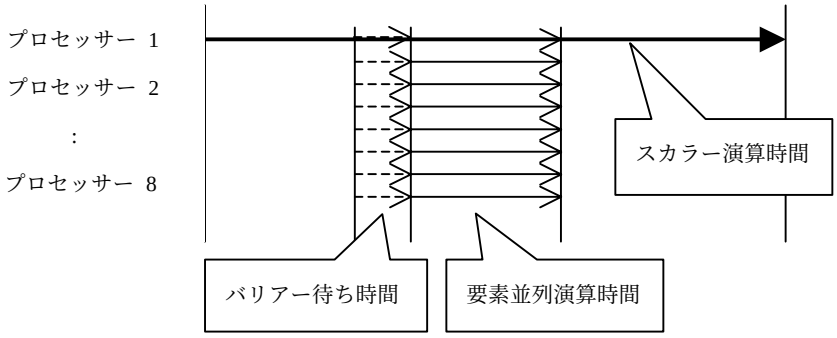
9.7. ジョブの開始、終了メール

ジョブが実行を開始､または終了したとき NQS はジョブの投入者(オプション#@$-muでメールアドレスを指定しているときは送信先）にメールを送信します。ジョブが実行を開始すると開始時のメールにはバッチキュー名や開始時刻が出力されますが、ジョブが実行されなかった場合には実行に失敗した理由等のエラーメッセージが出力されます。また、終了時のメールには以下のようなジョブの統計情報が出力されます。

submit user	ジョブをサブミットしたユーザー名
account number	ジョブの支払コード
request id	リクエスト ID（ジョブ ID）
submit host	ジョブをサブミットしたホスト名
execute host	サブミットしたジョブを実行したホスト名
submit queue	ジョブをサブミットしたキュー名
submit time	ジョブのサブミット時間
request start time	ジョブ開始時刻
request finish time	ジョブ終了時刻
request end status	ジョブの終了状態
request exit code	ジョブの返却コード
request cpu time (user + system)	ユーザー cpu 時間とシステム cpu 時間の合計
request exist time	ジョブ経過時間
used ES size	拡張記憶（ES）の最大使用量
dynamic allocated memory size	実メモリの平均使用量
number of forked process	フォークしたプロセス数
number of nodes	確保したノード数
number of threads	生成したスレッド数
number of EP processes	要素並列プログラムを実行したプロセス数
total computing time of EP processes	要素並列プログラムの総演算時間
scalar computing time of EP processes	要素並列プログラムのスカラー演算時間
parallel computing time of EP processes	要素並列プログラムの要素並列演算時間
barrier waiting time of EP processes	要素並列プログラムのバリアー待ち時間
number of total instructions	全命令実行回数
number of floating point operations	浮動小数点演算実行回数
floating point operations per second	浮動小数点演算実行回数 / ユーザー cpu 時間

要素並列プログラムの各時間には以下の関係があります。

要素並列プログラムの総演算時間 =（スカラー演算時間 －要素並列演算時間）+ 要素並列演算時間×8



9.8. 実行例

FORTRAN プログラムをコンパイルした後、NQS を使用してバッチ処理したときの実行例は以下のようになります。

ディレクトリーは **test** （**/home/**ログイン名**/test**）で作業します。

```
% cd test
```

ソースプログラムは以下の **test.f** を使用します。

```
% cat test.f
      parameter (n=10)
      dimension a(n,n),x(n),b(n)
      read(*,*)  ((a(j,i),j=1,n),i=1,n)
      read(*,*)  (x(j),j=1,n)
      do 10 i=1,n
       b(i)=0
 10   continue
      do 20 i=1,n
       do 30 j=1,n
        b(i)=b(i)+a(j,i)*x(j)
 30    continue
 20   continue
      do 40 i=1,n
       write(*,*) b(i)
 40   continue
      stop
      end
```

コンパイラーは **f77** を使用、ロードモジュール名を **program** とします。

```
% f77 test.f -o program
f77: compile start : test.f

*OFORT77 V01-03-/A entered.
*program name = MAIN
*end of compilation : MAIN
*program units = 0001, no diagnostics generated.
```

エラーはなく、ロードモジュール **program** が作成されました。

```
% ls
data     go.csh   program  test.f
```

スクリプトファイルは **go.csh** を使用します。（**go.csh** と **data** は予め用意しています。）

```
% cat go.csh
#!/bin/csh
#@$-q single                ≫スカラージョブなのでキューは single
#@$-lT 1:00:00              ≫実行時間は 1 時間
#@$-lM 100MB                ≫メモリー使用量は 100MB
cd test                     ≫作業ディレクトリーtest へ移動
program < data              ≫プログラムの実行（ファイル data は入力データー）
%
```

スクリプトファイル **go.csh** をサブミットします。

```
% qsub go.csh
Request 10627.mpp-s.cc.u-tokyo.ac.jp submitted to queue: single.
```

ジョブの実行状況は **qstat** コマンドで確認できます。

```
% qstat
 （省略）
2001/08/13 (Mon) 11:27:13:   BATCHPIPE REQUESTS on mpp-bt.cc.u-tokyo.ac.jp
NQS schedule stop time : 2001/08/18 (Sat) 18:00:00 (Remain:  126h 32m 47s)
   REQUEST      NAME      OWNER      QUEUE     PRI NICE   CPU    MEM    STATE
10627.mpp-s.cc  go.csh    a30000     C         31  20     28800  13824  RUNNING
```

実行が終了すると結果のファイル（この例では **go.csh.e10627**、**go.csh.o10627**）が作成されます。

```
% ls
data           go.csh.e10627  program
go.csh         go.csh.o10627  test.f
% cat   go.csh.o10627
  0.0
   1.00000000
   2.00000000
   3.00000000
   4.00000000
   5.00000000
   6.00000000
   7.00000000
   8.00000000
   9.00000000
```

特にエラーがなければエラー出力ファイルは空です。

```
% cat go.csh.e10627
%
```

NQS からのメールも到着しています。（1 通目は開始時メール、2 通目が終了時メール）

```
% mail
mailx [5.2 UCB] [OSF/1]  Type ? for help.
"/var/spool/mail/a30000": 2 messages 2 new 2 unread
>N  1 root@mpp-bt.cc.u  Mon Aug 13 11:26  23/1039 "NQS Initiator Report: 10627."
 N  2 root@mpp-bt.cc.u  Mon Aug 13 11:26  43/2163 "NQS Terminator Report: 10627"
? 2
Message  2:
From daemon Mon Aug 13 11:26:55 2001
Date: Mon, 13 Aug 2001 11:27:14 +0900 (JST)
From: system root account <root@mpp-bt.cc.u-tokyo.ac.jp>
To: a30000@mpp-s.cc.u-tokyo.ac.jp
Subject: NQS Terminator Report: 10627.mpp-s.cc.u-tokyo.ac.jp

submit user                             = a30000
account number                          = A(default)
request id                              = 10627.mpp-s.cc.u-tokyo.ac.jp
submit host                             = mpp-s.cc.u-tokyo.ac.jp
execute host                            = mpp-bt.cc.u-tokyo.ac.jp
submit queue                            = C
submit time                             = 2001/08/13 (Mon) 11:26:33
request start time                      = 2001/08/13 (Mon) 11:26:55
request finish time                     = 2001/08/13 (Mon) 11:27:03
（省略）
? q
%
```

☞ 上記の例ではプログラム（**test.f**）、スクリプトファイル（**go.csh**）のほか、入力データー**(data)**が必要です。エディターなどで予め作成して下さい。

```
% vi data
1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
0 1 2 3 4 5 6 7 8 9
```