汎用UNIXサーバにおけるプログラミング言語利用法

渡部 善隆 *

2000 年 1 月より，汎用計算機 FUJITSU M-1800/20U が汎用 UNIX サーバ FUJITSU GP7000F モデル 900 に置き換えられました． UNIX システムのホスト名は同じ “kyu-cc” です．旧 kyu-cc の利用者ファイルはそのまま新 kyu-cc に引き継がれています．

新 kyu-cc では，利用者が確保できる最大記憶容量が旧 kyu-cc の 0.5GB から **32GB へと飛躍的に増大**しました．また，いくつかの Fortran プログラムを用いて処理速度を比較したところ，M-1800/20U のベクトルユニットを使用しない場合の計算において新 kyu-cc は旧 kyu-cc の **2 倍から 5 倍の性能**を示しました．この「2 倍から 5 倍」という性能は，新 kyu-cc のひとつの CPU における測定結果です．新 kyu-cc では，さらに自動並列化オプション，最適化指示行， OpenMP Fortran， MPI，並列版数値計算ライブラリなどによる並列化手法を用いることで，最大 32CPU を使った計算を行なうことができます．つまり，理論上は**さらに 32 倍速くなる**可能性があります．

この記事では，新 kyu-cc のプログラム言語である Fortran，C，C++ の基本的な利用方法について説明します．なお， kyu-cc の利用に関する最新の情報，オンラインマニュアルへのリンクなどは

`http://www.cc.kyushu-u.ac.jp/library/GP7000F/GP7000F.html`

で適宜お知らせします．

1 汎用 UNIX サーバ

ここでは，汎用 UNIX サーバ FUJITSU GP7000F モデル 900 のプログラミング言語を利用するために必要な基本的な事項を説明します．

1.1 kyu-cc

ネットワーク経由で汎用 UNIX サーバに接続する場合のホスト名，ドメイン名， IP アドレスは表 1 の通りです．

表 1: 汎用 UNIX サーバのホスト名など

ホスト名	kyu-cc
ドメイン名	cc.kyushu-u.ac.jp
IP アドレス	133.5.9.1

ssh, telnet, rlogin などで kyu-cc に接続する場合には，ホスト名 + ドメイン名の “ kyu-cc.cc.kyushu-u.ac.jp ” または IP アドレスの “ 133.5.9.1 ” を指定します．具体的な接続方法，ファイルの転送方法は [20], [21], [22] を参照してください．

1.2 オペレーティングシステム

kyu-cc のオペレーティングシステム (OS) は，研究室のワークステーションなどで広く用いられている UNIX OS の一種である Solaris7 です． UNIX を使ったことのある方ならば，違和感なく kyu-cc を利用することができます．日本語文字コードは EUC，ログインシェルは csh です． UNIX の基本的な利用方法については [23] を参照してください．

*九州大学大型計算機センター E-mail: watanabe@cc.kyushu-u.ac.jp http://www.cc.kyushu-u.ac.jp/RD/watanabe/

1.3 kyu-vpp, wisdom とのファイル共有

kyu-cc のホームディレクトリにある“ VPP ”という名前のディレクトリとスーパーコンピュータ FUJITSU VPP700/56 (ホスト名: kyu-vpp) のホームディレクトリとの間にシンボリックリンクが張られています．VPP 配下のファイルを指定することにより，kyu-vpp のファイルを kyu-cc のアプリケーション・ソフトウェアの入力データとしたり，kyu-vpp のファイルを編集し，バッチジョブを投入することもできます．

また，端末サーバ wisdom からは kyu-cc, kyu-vpp のファイルを読み書きすることができます．これまで wisdom で動作していた SPSS, *Mathematica*, LaTeX 2εなどのソフトウェアは 2000 年 1 月より kyu-cc に移行されています．利用者ファイルは wisdom にそのまま残されていますので，必要なファイルは適宜 kyu-cc へコピーしてください．

1.4 エディタ

現在 kyu-cc で利用できるエディタは表 2 の通りです． mule の使用方法は [23] を， je の利用方法は [24] を参照してください．

表 2: kyu-cc のエディタ

エディタ名	コマンド名	備考
mule	`mule(/usr/local/bin/mule)`	mule 2.3．日本語入力は Free Wnn 1.1
vi	`vi(/usr/bin/vi)`	利用方法は `man vi`
je	`je(/usr/local/bin/je)`	PFD ライクなエディタ

1.5 実行の手順

Fortran, C, C++ で記述したプログラムを動作させるには，翻訳・結合編集・実行という 3 つの過程を踏みます．

最初の翻訳処理[1]により，プログラミング言語を計算機の理解できる機械語に変換します．このファイルを**オブジェクトファイル**と呼びます．数値計算ライブラリである SSL II([7]), IMSL ライブラリ ([14]) などはオブジェクトファイルの集合体です．

オブジェクトファイルはそのままでは実行できません．実行のために必要な手続きをオブジェクトファイルに組み込む処理を**結合編集**[2]と呼びます．結合編集処理により作成されたファイルを**実行可能ファイル**と呼びます．実行可能ファイル名をコマンドとして指定することによって利用者がソースプログラムに記述した処理が実行されます．

次章で説明するように，通常 `frt`, `fcc`, `FCC` コマンドを使用すると，ソースプログラムを翻訳し結合編集により実行可能ファイルを作成するまでの手順が自動的に行われます．実行可能ファイルが作成された時点で中間ファイルであるオブジェクトファイルは消去されます．オプションを指定することで，翻訳段階で処理を打ち切り，オブジェクトファイルを保存することもできます．保存されたオブジェクトファイルは，結合編集の際に必要に応じてメインプログラムから呼び出されることで，実行ファイルを構成する一部となります (図 1)．

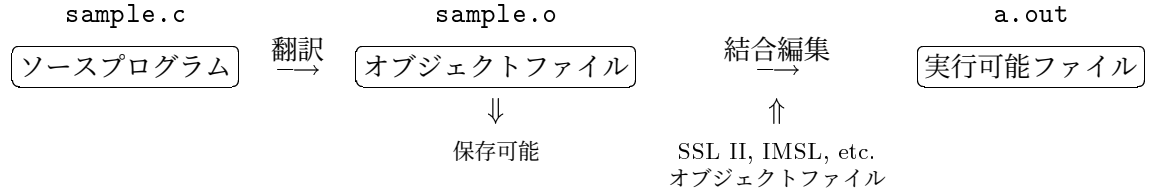

図 1: 実行可能ファイルの作成手順

[1]「コンパイル」と呼ぶこともあります．

[2]「リンク・エディット」または単に「リンク」と呼ぶこともあります．

1.6 利用形態

kyu-cc でプログラムを翻訳・結合編集・実行する方法は，**対話型処理**と**バッチ処理**に分かれます．

対話型処理とは，コマンド入力によって翻訳や実行などをインタラクティブに行なう利用形態のことです．対話型処理は翻訳やデバッグ，すぐに結果が返ってくるようなプログラムの実行に適しています．

バッチ処理とは，一連の処理手順をシェルスクリプトファイルに記述した「バッチリクエスト」と呼ばれるものを `qsub` コマンドによって計算機に依頼する利用形態のことです．バッチ処理は多くの記憶領域や CPU を必要としたり，処理時間が数十分，数時間におよぶプログラムの実行に適しています．15 分を超えるバッチ処理については，対話型処理に比べ 格段に安い 利用負担金が設定されています[3]．長時間におよぶジョブは，是非バッチ処理で実行してください．

なお UNIX でよく行なう，コマンドラインの最後に “ `&` ” をつける処理方法は**バッチ処理ではなく対話型処理とみなされます**ので注意してください．

1.7 制限値

対話型処理・バッチ処理の制限値は表 3 の通りです．

表 3: kyu-cc の制限値

処理形態	キュー名	記憶容量	CPU 時間 *	備考
対話型処理	—	1GB	12 時間	12CPU まで使用可
バッチ処理	sc	4GB	120 時間	非並列向け
	sc8	8GB	120 時間	8CPU まで使用可
	sc32	32GB	120 時間	32CPU まで使用可

* 並列ジョブは各 CPU 時間の合計

2 Fortran の対話型処理

ここでは， Fortran プログラムの対話的な利用方法について説明します．詳しい利用方法は [1] を，また Fortran の文法については [2] を参照してください

2.1 旧 kyu-cc からの変更点

おもな変更点は以下の通りです．

1. 従来の FORTRAN 77 から Fortran 95 規格に完全準拠となりました．ほとんどの利用者プログラムは修正することなく新 kyu-cc でも翻訳可能です．オブジェクトファイル，実行可能ファイルは再作成が必要です．詳しくは [1] の「富士通ホスト系 FORTRAN コンパイラとの互換」を参照してください．

2. 浮動小数点形式が IBM 形式 (M 形式) から kyu-vpp と同じ IEEE 形式に変更されました．詳しくは [1] の「IBM370 形式 –IEEE 形式浮動小数点データ入出力変換」を参照してください．

3. 実行可能ファイルに先だって “ `./` ” の指定が必要になりました．

4. SSL II の結合方法が “ `-lssl2` ” または “ `-lssl2vp` ” から “ `-lfssl2` ” に変更になりました．

5. 最適化に関するオプション，デバッグオプションが変更されています．

[3] 2000 年 2 月現在の kyu-cc の演算負担金は，対話型処理が 1 秒につき 1 円，バッチ処理が 5 分まで 1 秒につき 2 円，5 分を超え 15 分まで 1 秒につき 1 円，15 分を超えると 1 秒につき 0.1 円です．利用負担金の詳細は `http://www.cc.kyushu-u.ac.jp/service/futankinhyo.html` を御覧ください．

2.2 コマンド名とファイル拡張子

kyu-cc の Fortran は，最新の JIS 規格 (JIS X 3000-1) に完全準拠した自動並列化機能を有する Fortran 95 コンパイラです． Fortran の翻訳と結合編集のコマンドは `frt(/opt/FSUNf90/bin/frt)` または `f90(/opt/FSUNf90/bin/f90)` または `f95(/opt/FSUNf90/bin/f95)` です． 3 つのコマンドは等価です． Fortran ソースファイルの拡張子[4]はプログラムの形式に応じて表 4 のようにしてください．

表 4: Fortran のファイル拡張子

拡張子	プログラムの種類
`.f90` または `.f95`	自由形式
`.f` または `.for`	固定形式

ソース形式の解釈は，翻訳時オプション `-Fixed` または `-Free` で変更することもできます．

2.3 基本的な手順

`frt`(または `f90`, `f95`) コマンドにより翻訳および結合編集を行い，実行可能ファイルを作成します．以下は，ファイル名を“ `example.f95` ”とした例です．

```
kyu-cc% frt example.f95 ↵          ←翻訳と結合編集により実行可能ファイルを作成
```

処理が重大なエラーを起こすことなく終了したならば，実行可能ファイル“ `a.out` ”が作成されています．実行はファイル名をコマンドとして入力します．

```
kyu-cc% ./a.out ↵                  ←実行
```

【注意】

`a.out` に先だって指定する“ `./` ”は，「現在作業しているディレクトリ (カレントディレクトリ) にあるファイルを対象にする」という意味です．旧 kyu-cc(FUJITSU M-1800/20U の UXP/M システム) では，“ `a.out` ”と指定するだけで実行することができました．しかし，セキュリティ強化[5]のため，新 kyu-cc においては利用者が設定を変更しない限りこのような実行はできないようになりました．必ず最初に“ `./` ”を指定してください．

2.4 翻訳・実行時の診断メッセージ

プログラムの翻訳または実行時に，たとえば以下のようなメッセージが出力されることがあります．

```
kyu-cc% frt jprmx.f ↵
Fortran diagnostic messages: program name(FMF001)
  jwd2006i-i  "jprmx.f", line 123: この名前は, 宣言だけされていて引用されていません.(名
前:VC)
  jwd2004i-i  "jprmxss.f", line 700: この変数は, 値を設定していますが引用されていません.(名
前:GKEY)
                :
```

[4] ファイル名の最後に置く文字列で，ファイルの属性や内容を表す目的で使われます．

[5] たまたま実行可能ファイルと同じ名前の (カレントディレクトリにない)UNIX コマンド，シェルスクリプトなどが呼び出されてしまい，最悪，大切なファイルが消去されてしまうなどの予期せぬ事態を防ぐことが目的です．また，現在のスーパーコンピュータ VPP700/56(ホスト名: kyu-vpp) では“ `./` ”の指定は必要ありませんが，来年 1 月導入予定の次期スーパーコンピュータでは設定を変更する予定です．

```
kyu-cc% ./a.out ↵
 jwe0011i-e A floating overflow exception was detected.
  error occurs at g_adxi   loc ff17692c offset 000001a8
  g_adxi (f)   at loc ff176784 called from loc 00035b54 in mpfunmod.mpinrl_
               :
```

“jwd” で始まるメッセージは翻訳時の診断メッセージです．最後の文字によってメッセージのレベルが判定できます．詳細は [3] を参照してください．

表 5: 翻訳時の診断メッセージの最後の文字と意味

文字	意味
i	エラーではないが注意を促すメッセージです．
w	軽度のエラーが発生したことを示します．翻訳処理は続行されます．
s	重度のエラーを示すメッセージであり，システムはエラーの文を無視し，その他の文については処理を続行します．
u	致命的なエラーを示すメッセージであり，システムは処理を打ち切ります．

“jwe” で始まるメッセージは実行時の診断メッセージです．こちらも最後の文字によってメッセージのレベルが判定できます．詳細は [4] を参照してください．

表 6: 実行時の診断メッセージの最後の文字と意味

文字	意味
i	エラーではないが注意を促すメッセージです．
w	軽度のエラーが発生したことを示します．実行は続行されます．
e	中度のエラーが発生したことを示します．エラーが発生した文は無視されます． 10 回エラーが発生すると実行が打ち切られます．
s	重度のエラーが発生したことを示します．実行は打ち切られます．
u	実行が続行できないような致命的なエラーが発生したことを示します．実行は打ち切られます．

2.5 オブジェクトファイルの作成と利用方法

翻訳時オプション -c を指定することにより，結合編集を行わず，オブジェクトファイルの作成までを行うこともできます．

```
kyu-cc% frt -c example.f95 ↵          ←オブジェクトファイルの作成
```

オブジェクトファイルの拡張子は .o です．例では “example.o” という名前のファイルが作成されます．

オブジェクトファイルは，既にデバックが終了した副プログラムをメインプログラムと切り離して管理する場合によく用いられます．例として，メインプログラムを “main.f95”， FORTRAN 77 で記述された副プログラムを記述したファイルを “sub.f” とします．まず，sub.f を -c オプションを付けて翻訳し，オブジェクトファイル sub.o を作成します．

```
kyu-cc% frt -c sub.f ↵                ←オブジェクトファイルの作成
```

次に，メインプログラムを翻訳し，オブジェクトファイルを結合することで実行可能ファイルを作成します．

```
kyu-cc% frt main.f95 sub.o ↵          ←実行可能ファイルの作成
```

このようにすると，メインプログラムを何度も修正する場合でも副プログラムの翻訳は一度だけで済むため，翻訳時間が短縮できます．

frt コマンドには上の例のようにオブジェクトファイル (拡張子 .o のファイル) も指定することができます．また，複数のソースプログラムファイル，オブジェクトファイルも指定することができます．

2.6 翻訳時オプション

よく用いる Fortran の翻訳時オプションを表 7 に示します．

表 7: よく用いる Fortran の翻訳時オプション

オプション	機能
-c	オブジェクトファイルの作成までを行います．結合編集を行わず実行可能ファイルは作成されません．
-o *filename*	実行可能ファイル名またはオブジェクトファイル名を *filename* に変更します．
-Free	自由形式の Fortran プログラムとして翻訳します．
-Fixed	固定形式の Fortran プログラムとして翻訳します．
-Hasu	プログラムのデバッグのため，引数の妥当性の検査，添字式・部分列範囲の検査，未定義データの引用の検査を行ないます．メッセージは翻訳時，実行時に出力されます．実行時間が増大するため，小規模なプログラムに対しデバッグを行ない，デバッグ終了後は必ず，実行可能ファイル，オブジェクトファイルは再作成してください．
-Am	モジュールを翻訳する場合に指定します．詳しくは [1] の「モジュールおよびモジュール引用の注意事項」を参照してください．
-Kfast	翻訳しているマシン上で高速に実行させることを指示します．
-Keval	演算評価方法の変更の最適化が行われます．計算誤差に影響が生じることがありますので注意が必要です．詳しくは [1] の「最適化機能」を参照してください．
-KV8PFMADD	GP7000F モデル 900 のアーキテクチャを活かしたオブジェクトプログラムを生成します．
-Kgs	広域命令スケジューリングによる最適化を指示します．
-Kprefetch	メモリの先読み機能を使用した最適化を指示します．
-X9	言語仕様が Fortran 95 であると解釈して翻訳します．拡張子が .f または .for のファイルに Fortran 90 から新たにサポートされた組込み関数などを記述する場合に必要です．
-KV9	64 ビットモードの実行可能プログラムを作成します．記憶容量が 2GB を超える場合には必ず指定してください．ただし，制限値の関係から対話型処理による実行はできません．-KV9 オプションを指定しないオブジェクトファイルとの混在はできません．

◇デバッグオプションの指定例

以下は，デバッグオプションを指定して翻訳・結合編集し，実行する例です[6]．

[6] 予期しないバグが見つかる場合もあります．筆者はプログラム開発の最終チェックとしてよく使います．

```
kyu-cc% frt -Hasu example.f95 ↵                ←デバッグオプションの指定
kyu-cc% ./a.out ↵                              ←実行を通してデバッグ
jwe0320i-w line 7 The subscript or substring M is out of the specified range
 (reference value: 334,1001, specification value: 1:1000,1:1000).
 error occurs at MAIN__   line 7 loc 00010a1c offset 00000180
 MAIN__       at loc 0001089c called from o.s.            ←警告メッセージ
taken to (standard) corrective action, execution continuing.
```

◇最適化オプションの指定例

-Kfast オプションと -Keval オプションを指定します．プログラムによってはかなりの高速化が得られる場合があります．ただし，翻訳時間は一般に増大します．

```
kyu-cc% frt -Kfast,eval main.f90 ↵      ←最適化オプションの指定
kyu-cc% ./a.out ↵                       ←実行
```

◇最大限の最適化オプション

メーカーに確認した最大限の最適化オプションは以下の通りです．ただし，実行結果に副作用が生じる可能性があります．また，他の SPARC マシンへの移植もできないため，指定する場合は注意してください．

```
kyu-cc% frt -Kfast,eval,V8PFMADD,gs,prefetch main.f90 ↵  ←最大限の最適化
```

◇環境変数による翻訳時オプションの設定

環境変数 FORT90C の値として翻訳時オプションを設定することができます．よく利用する翻訳時オプションを設定しておくと便利です．以下は，翻訳時オプション -Kfast,eval を設定し，翻訳・結合編集・実行を行なった後，設定を解除する例です．

```
kyu-cc% setenv FORT90C '-Kfast,eval' ↵     ←環境変数の設定
kyu-cc% printenv FORT90C ↵                 ←設定された値の確認 (printenv コマンド)
-Kfast,eval
kyu-cc% frt leq.f ↵                        ←翻訳・結合編集 (-Kfast,eval は有効)
kyu-cc% ./a.out ↵                          ←実行
          :
kyu-cc% unsetenv FORT90C ↵                 ←設定の解除
kyu-cc% printenv FORT90C ↵                 ←値の確認
kyu-cc%                                    ←何も設定されていない
```

2.7　実行時オプション

実行時オプションは実行可能ファイル名の後に“ -Wl, ”(“ l ”は英小文字) を指定し，続いてオプションを指定します．複数の実行時オプションはカンマ（,）で区切って続けます．

よく用いる実行時オプションを表 8 に示します．

表 8: よく用いる Fortran の実行時オプション

オプション	機能
-u	浮動小数点アンダーフローが発生した場合，エラーメッセージを出力します.
-C*number*	書式なし入出力文において，装置番号 *number* から IBM370 形式浮動小数点データのファイルを入出力するときに指定します． *number* を省略した場合すべての装置番号に対して有効となります.
-M	-C の指定により行なわれる IBM370 形式から IEEE 形式への浮動小数点データの変換によって仮数部のビットが損失したときに，診断メッセージを出力します.
-T*number*	書式なし入出力文において，装置番号 *number* からリトルエンディアンデータのファイルを入出力するときに指定します． *number* を省略した場合すべての装置番号に対して有効となります.

以下は，実行時オプション -C を指定して装置番号 10 番から IBM370 形式浮動小数点データのファイル入出力を行ない，あわせて診断メッセージを出力する例です．旧 kyu-cc の書式なし WRITE 文で出力したデータファイルを読み込む場合は，このオプションを指定してください.

```
kyu-cc% ./a.out -Wl,-C10,-M ↵      ← IBM370 形式浮動小数点データのファイル入出力
```

2.8 時間計測コマンド

timex(/usr/bin/timex) コマンドにより，翻訳・結合編集・実行に要する経過時間， CPU 時間を計測することができます.

```
kyu-cc% timex frt example.f95 ↵          ←翻訳・結合編集時間の計測
real        4.01                         ←経過時間 (4 秒 01)
user        1.70                         ←ユーザー CPU 時間
sys         0.80                         ←システム CPU 時間

kyu-cc% timex ./a.out ↵                  ←実行時間の計測
real       10.30                         ←経過時間 (10 秒 30)
user       10.24                         ←ユーザー CPU 時間
sys         0.03                         ←システム CPU 時間
```

また， Fortran 95 標準の組込みサブルーチンとして，経過時間を計測する SYSTEM_CLOCK， CPU 時間を計測する CPU_TIME が利用できます．これらのサブルーチンを拡張子 .f または .for のついたファイルに記述した場合には，翻訳時オプション -X9 が必要です．利用方法はコマンドラインから man system_clock またはman cpu_time で調べることができます.

2.9 数値計算・図形ライブラリの結合

数値計算ライブラリ，図形ライブラリを使用する場合には，frt(または f90, f95) コマンドのオプション -l に続けてライブラリ名を指定します．-l は結合編集時オプションのため，次の例のように，コマンド並びの 最後 に指定してください.

```
kyu-cc% frt main.f90 -lfssl2 ↵          ← SSL II の結合
```

kyu-cc で利用できるライブラリと結合方法などは表 9 の通りです． SSL II の指定方法が -lssl2 から -lfssl2 に変更されました[7].

[7] -lssl2 と指定すると， Sun SPARC 版の SSL II が起動され，結合編集に失敗します.

利用方法は参考文献を参照してください.

表 9: 数値計算・図形ライブラリ

ライブラリ名	結合方法	文献
SSL II	`-lfssl2`	[7], [8], [9]
NUMPAC	`-lnumpac`	[11]
BLAS	`-lblasmt`	[12]
LAPACK	`-llapackmt -lblasmt`	[12]
ScaLAPACK	`-lscalapack`	[12]
NAG Fortran SMP Library	(`nagfsmp` コマンド)	[13]
NAG Fortran 90 Library	(`nagf` コマンド)	[13]
IMSL Fortran 90 MP Library	(`imslfmpi` コマンド)	[14]
CalcPS	`-lcalcps`	[15]

2.10 ファイル入出力

ここでは, OPEN 文にファイル名を指定しない場合のファイル入出力について簡単に説明します.

◇標準入出力

装置番号 5 番 (READ(5) に対応, 標準入力と呼びます) と 6 番 (WRITE(6) に対応, 標準出力と呼びます) は通常端末の入出力になります. UNIX の「リダイレクション機能」を用いることで, これらの入出力をファイルに切替えることができます. ただし, 操作に慣れるまでは, リダイレクションの方向に十分注意するようにしてください[8]. 以下の例では, 実行可能ファイルを “ `a.out` ”, 入力ファイルを “ `in.data` ”, 出力ファイルを “ `out.data` ” としています.

```
kyu-cc% ./a.out < in.data ↵      ←装置番号 5 番から in.data の内容を読み込む
```

```
kyu-cc% ./a.out > out.data ↵     ←装置番号 6 番への出力結果を out.data に保存 (上書き)
```

```
kyu-cc% ./a.out >> out.data ↵    ←装置番号 6 番への出力結果を既存の out.data に 追加 書き
```

```
kyu-cc% ./a.out >& out.data ↵    ←出力結果および実行時メッセージを out.data に保存
```

```
kyu-cc% ./a.out < in.data > out.data ↵ ← in.data から読み込み, 結果を out.data に保存
```

◇環境変数による結合

標準入出力以外の装置番号とファイルとの結合は, 環境変数 “ `fu`*xx* ” で行ないます. *xx* に装置番号を指定します. 番号が一桁の場合 “ `fu03` ” などと指定してください. 環境変数を設定せずにファイル出

[8] リダイレクションはたいへん便利な UNIX の機能です. しかし, 方向 (<, >) を間違えると大切なファイルが上書きされてしまう危険があります. 危険防止のために, 重要な入力ファイルは書き込み権を `chmod` コマンドで除去しておくか, コピーをとっておくことをお勧めします.

力を行なった場合は，“ fort.*xx* ”というファイルに出力されます．標準入力以外の装置番号に対し環境変数を設定せずにファイル入力を行なった場合には実行時にエラーとなります．

以下の例は，装置番号 10 番とファイル“ example.data ”を結合し，実行後，unsetenv によって結合を解除する例です．

```
kyu-cc% setenv fu10 example.data ↵          ←ファイルと装置番号の結合
kyu-cc% ./a.out ↵                          ←実行
      :
kyu-cc% printenv fu10 ↵                    ← printenv による確認
example.data
kyu-cc% unsetenv fu10 ↵                    ←結合の解除
```

2.11 自動並列化

kyu-cc の Fortran において複数の CPU を用いた並列処理を行う方法は表 10 の通りです．

表 10: kyu-cc における並列化の手段

並列化の手段	参考文献
自動並列化オプション	[1]
自動並列化オプション + 並列化制御行	[1]
OpenMP Fortran	[16]
MPI	[17]

ここでは，もっとも簡単な並列化の方法である自動並列化オプションの指定方法について説明します．なお， 2000 年 2 月現在，最大利用可能 CPU 台数は対話的処理では 12，バッチ処理では 32 です．

【注意】

自動並列化の対象は，コンパイラが並列化可能であると判断できる DO ループと配列操作の文に限られます．したがって，すべての Fortran プログラムが自動並列化によって高速に実行できるとは限りません．

FUJITSU GP7000F モデル 900

◇自動並列化オプション

自動並列化に関する翻訳時オプションは表 11 の通りです．表中の「リダクション」とは，総和，最大値などのような配列データからスカラー値を求める演算のことです[9]．

表 11: 自動並列化に関する翻訳時オプション

オプション	機能
-Kparallel	自動並列化を行います．
-Kreduction	リダクションによるループスライスを行います．並列化が促進される場合がある一方で結果に精度差が生じる可能性があります．
-Kinstance=n	CPU を n 台利用することを指定します． n は実行環境の CPU 数以下でなければなりません．また，実行時に指定する環境変数 PARALLEL に設定した値と異なる場合には動作が保証されません．-Kparallel オプションが指定されていない場合は無効です．
-Kocl	自動並列化を促進する制御行を有効にします．-Kparallel オプションが指定されていない場合は無効です．
-Et	自動並列化された DO ループを翻訳時に通知します．

-K で始まるオプションは，カンマで区切って続けて，例えば“ -Kparallel,reduction,instance=4 ”のように指定できます．

◇環境変数の設定

実行にあたり，環境変数 PARALLEL によって並列に動作させる CPU 数を設定します．必ず翻訳時オプション“ -Kinstance=n ”で指定した数と同じ値を指定してください．

【注意】

現在の kyu-cc の負担金体系では，プログラムを並列実行した場合，その合計の CPU 時間が課金対象になります．したがって，並列化性能の高くないプログラムを自動並列化した場合，経過時間はそれほど変わらないのに大幅に課金が増える可能性があります．このため，必ず利用者自身で -Kinstance オプションと環境変数 PARALLEL を同時に指定して使用 CPU 台数を管理するとともに，自分のプログラムがどの程度自動並列化による性能向上が得られるのかを把握してください．

◇自動並列化の実行例

- “ test.f ”を自動並列化オプションを指定し，翻訳・結合編集・実行します．並列化にあわせてリダクションによる最適化を指示します． CPU は 4 台用います．

```
kyu-cc% frt -Kparallel,reduction,instance=4 test.f ↵   ←自動並列化
kyu-cc% setenv PARALLEL 4 ↵                            ←環境変数の設定
kyu-cc% ./a.out ↵                                      ←実行
```

  instance と PARALLEL で指定する CPU 数は必ず同じ値を指定してください．

- “ test2.f ”を自動並列化オプションを指定し， 翻訳・結合編集・実行します．並列化にあわせてリダクションによる最適化を指示します．また，-Et オプションにより並列化されたことを確認します． CPU は 8 台用います．

[9] たとえば内積計算では，2 つのベクトルから 1 つのスカラー値を計算します．総和演算は単純に並列化できません．しかし，足し込み順序を変更できれば，各 CPU が割り当てられた部分和を計算して最後にそれぞれの部分和を集めることで並列化可能です．ただし，演算順序を変更することにより，丸め誤差の影響を受ける場合もあります．

```
kyu-cc% frt -Kparallel,reduction,instance=8 -Et test2.f ↵    ←自動並列化
kyu-cc% setenv PARALLEL 8 ↵                                  ←環境変数の設定
kyu-cc% ./a.out ↵                                            ←実行
```

- “ test3.f95 ”を自動並列化オプションを指定し，翻訳・結合編集・実行します．さらに，最大限の最適化を施します．実行結果に副作用が生じる可能性があるため注意が必要です． CPU は 8 台用います．

```
kyu-cc% frt -Kparallel,instance=8,fast,eval,V8PFMADD,gs,prefetch \ ↵
? test3.f95 ↵                                                ←自動並列化 + 最適化
kyu-cc% setenv PARALLEL 8 ↵                                  ←環境変数の設定
kyu-cc% ./a.out ↵                                            ←実行
```

  最初のコマンドラインのバックスラッシュ (\) は，コマンドの継続を意味します[10]．-Kparallel オプションと -Keval オプションが同時に指定された場合，-Kreduction オプションが指定されたとみなされます．

◇最適化制御行

ソースプログラムに最適化制御行を挿入することで自動並列化を促進することができます．最適化制御行を有効にするためには翻訳時オプション -Kparallel,ocl を指定します．詳しくは [1] を参照してください．

3 C, C++ の対話型処理

ここでは， C および C++ プログラムの対話的な利用方法について説明します．詳しい利用方法は [5], [6] を参照してください．

3.1 コマンド名とファイル拡張子

C の翻訳と結合編集のコマンドは fcc(/opt/FSUNf90/bin/fcc) です． C++ の翻訳と結合編集のコマンドは FCC(/opt/FSUNf90/bin/FCC) です． UNIX は大文字と小文字を区別します．ソースファイルの拡張子は表 11 の名前にしてください．

表 11: C, C++ のファイル拡張子

拡張子	言語名
.c	C または C++
.cc	C++
.C	C++

3.2 基本的な手順

C プログラムは fcc コマンドにより翻訳および結合編集を行い，実行可能ファイルを作成します．以下は，ファイル“ example.c ”から実行可能ファイルを作成する例です．

[10]原稿のスペースが足りないための便宜的なものです．必ずしも継続する必要はなく，もちろん一行で入力して構いません．

```
kyu-cc% fcc example.c ↵      ←翻訳と結合編集により実行可能ファイルを作成 (C の場合)
```

処理が正常に終了した場合，実行可能ファイル“ a.out ”が作成されています． 実行はファイル名をコマンドとして入力します．

```
kyu-cc% ./a.out ↵            ←実行
```

a.out に先だって指定する“ ./ ”の意味，注意事項は 2.3節の「注意」を参照してください．

C++ プログラムは FCC コマンドにより翻訳および結合編集を行い，実行可能ファイルを作成します．以下は，ファイル“ example.C ”から実行可能ファイルを作成する例です．

```
kyu-cc% FCC example.C ↵      ←翻訳と結合編集により実行可能ファイルを作成 (C++ の場合)
```

3.3 オブジェクトファイルの作成と利用方法

翻訳時オプション -c を指定することにより，結合編集を行わず，オブジェクトファイルの作成までを行うこともできます．具体的な手順は Fortran の対話型処理の 2.5節を参照してください．コマンド名とファイル拡張子を読み替えることにより，同様の処理が可能です．ただし， Fortran と C, C++ との間のオブジェクトファイルの結合はこの手順ではできません．言語間の結合方法は [1] の「言語間結合」の章を参照してください．

3.4 翻訳時オプション

よく用いる C, C++ の翻訳時オプションを表 12 に示します．詳細は [5], [6] を参照してください．

表 12: よく用いる C, C++ の翻訳時オプション

オプション	機能
-c	オブジェクトファイルの作成までを行います．結合編集を行わず実行可能ファイルは作成されません．
-o *filename*	実行可能ファイル名またはオブジェクトファイル名を *filename* に変更します．
-Kfast	翻訳しているマシン上で高速に実行させることを指示します．
-Keval	演算評価方法の変更の最適化が行われます．実行結果に副作用を生じることがあるため注意が必要です．
-KV8PFMADD	GP7000F モデル 900 のアーキテクチャを活かしたオブジェクトプログラムを生成します．
-Kgs	広域命令スケジューリングによる最適化を指示します．
-Kprefetch	メモリの先読み機能を使用した最適化を指示します．
-KV9	64 ビットモードの実行可能プログラムを作成します．記憶容量が 2GB を超える場合には必ず指定してください．ただし，制限値の関係から対話型処理による実行はできません．-KV9 オプションを指定しないオブジェクトファイルとの混在はできません．

◇最適化オプションの指定例

-Kfast オプションと -Keval オプションを指定します．プログラムによってはかなりの高速化が得られる場合があります．ただし，翻訳時間は一般に増大します．

```
kyu-cc% fcc -Kfast,eval example.c ↵     ←最適化オプション-Kfast, -Keval の指定
kyu-cc% ./a.out ↵                      ←実行
```

◇最大限の最適化オプション

メーカーに確認した最大限の最適化オプションは以下の通りです．ただし，実行結果に副作用が生じる可能性があります．また，他の SPARC マシンへの移植もできないため，使用する場合は注意してください．

```
kyu-cc% fcc -Kfast,eval,V8PFMADD,gs,prefetch main.c ↵      ←最大限の最適化
```

3.5 数値計算ライブラリの組み込み

C, C++ プログラムから利用できる数値計算ライブラリは表 13 の通りです．利用方法は参考文献を参照してください．

表 13: 数値計算ライブラリ

ライブラリ名	結合方法	文献
C-SSL II	`-lcssl2mt -lfssl2`	[10]
IMSL C Library	(`imslc` コマンド)	[14]

3.6 並列化

現在のところ， C, C++ の自動並列化機能はサポートされていません． C, C++ プログラムを並列化する手段として， MPI ライブラリを利用する方法と POSIX 規格に準拠したスレッドライブラリによる方法があります．利用方法は [17], [18], [19] を参照してください．

4 バッチ処理

Fortran, C, C++ プログラムの規模が対話型処理の制限値を超えた場合や実行が長時間にわたるジョブはバッチ処理により実行します．バッチ処理は対話型処理に比較して負担金が安価に設定されています．

この章では，バッチ処理の方法について説明します．

4.1 バッチ処理に用いるコマンド

バッチ処理は，対話型処理で行う一連の処理の流れを「バッチリクエスト」と呼ばれる シェルスクリプトに記述し，`qsub` コマンドによって投入します．バッチ処理に用いるコマンドは表 14 の通りです．

表 14: バッチ処理に用いるコマンド

コマンド	機能
`qsub` *options filename*	*filename* に記述したバッチリクエストを投入
`qstat`	バッチキューの状態表示
`qps`	投入したジョブの実行状況を表示
`qdel` *options request-ID*	バッチリクエスト識別子 *request-ID* をキャンセル

4.2 バッチリクエストの書き方

バッチリクエストはエディタを用いてファイルとして作成します．ここでは“ a.sh ”というファイル名としています．以下，具体的な例にそって説明します．

```
#                          ← csh であることを指定
cd EXAMPLE                 ←ディレクトリの移動
f90 main.f90               ←プログラムの翻訳
./a.out                    ←実行
```

- 先頭の“ # ”はジョブスクリプトを csh で記述することを指定します．# を指定しない場合は sh での記述になります． csh と sh では，環境変数の設定方法などが異なります．この記事では csh の記述にしたがっています[11]．また， mule, emacs でファイル拡張子“ .sh ”を付けて新規にファイルを作成すると，自動的に“ #!/usr/bin/csh ”が先頭に挿入されています．この記述でも問題ありません．
- 次は cd コマンドでディレクトリ“ EXAMPLE ”に移動しています．バッチリクエストの開始は利用者の**ホームディレクトリ (login 時のディレクトリ) から**になります．プログラムファイル，実行可能ファイルなどのあるディレクトリまで必ず移動するようにしてください．
- 以降の記述は対話型処理と同様の手順を記述します．ここでは“ main.f90 ”を翻訳・結合編集し，実行しています．

4.3 バッチリクエストの記述例

例 1. 既に対話型処理などで作成済みの実行可能ファイル“ a.out ”を実行します．作業用ディレクトリは“ mydir ”です．

```
#
cd mydir
./a.out
```

例 2. 記憶領域の容量が 2GB を超える Fortran プログラム“ main.f90 ”を翻訳・結合編集し，実行します．必ず，翻訳時オプション -KV9 を指定してください．-KV9 を指定して翻訳しないで作成したオブジェクトファイルとの互換性はありませんので注意してください．

```
#
cd mydir
f90 -KV9 main.f90
./a.out
```

例 3. Fortran プログラム“ main.f90 ”を翻訳・結合編集し，実行します．装置番号 5 番からの入力ファイルとして“ in.data ”を指定します．

```
#
cd mydir
f90 main.f90
./a.out < in.data
```

例 4. Fortran プログラムを翻訳・結合編集し，実行します．結合編集時にサブルーチンライブラリ SSL II([7]) を結合しています．装置番号 1 番， 23 番からの入出力ファイルに“ inout1.data ”，“ inout2.data ”をそれぞれ指定します．環境変数の設定は実行の前に行なってください．

[11] よくわからない方は，「おまじない」だと思って必ず先頭に記述してください．

```
#
cd mydir
f90 main.f90 -lfssl2
setenv fu01 inout1.data
setenv fu23 inout2.data
./a.out
```

例 5. 最適化オプションをいくつか指定して Fortran プログラムを翻訳・結合編集し，実行します．またその際 timex コマンドによって経過時間， CPU 時間を計測します．実行速度には影響ありません．

```
#
cd mydir
timex f90 -Kfast,eval,gs main.f90
timex ./a.out
```

例 6. C プログラム “ main.c ” を翻訳・結合編集し，実行します．標準入力ファイルとして “ in.data ”，標準出力ファイルとして “ out.data ” を指定します．

```
#
cd mydir
fcc main.c
./a.out < in.data > out.data
```

例 7. C++ プログラム “ main.C ” を翻訳・結合編集し，実行します．最適化オプション -Kfast,eval を指定しています．

```
#
cd mydir
FCC -Kfast,eval main.C
./a.out
```

例 8. 自動並列化・最適化オプションを指定して Fortran プログラムを翻訳・結合編集し，実行します．使用 CPU 数は 8，実行可能ファイル名を “ b.out ” に変更しています．

```
#
cd mydir
f90 -Kparallel,reduction,instance=8,fast,eval -o b.out main.f90
setenv PALALLEL 8
./b.out
```

例 9. “ #@$ ” に続けて，後述する qsub コマンドのオプションを指定することができます．この例では， sc8 キューに投入することを指示する -q sc8 オプションを記述しています．

```
#
#@$-q sc8
cd mydir
f90 -Kparallel,reduction,instance=8,fast,eval -o b.out main.f90
setenv PALALLEL 8
./b.out
```

例 10. あらかじめ作業を行なうディレクトリに “ a.out ” というファイルがある場合，そのファイルを消去してから翻訳・結合編集し，実行します． この処理によって，翻訳が正常に終了しなかった場合に既存の実行可能ファイルが実行されてしまうという事態を回避します．

```
#
cd mydir
if ( -f a.out ) then
  rm -f a.out
endif
f95 main.f95
./a.out
```

4.4 バッチリクエストの投入

バッチリクエストの投入は qsub(/usr/bin/qsub) コマンドによって行ないます．投入するジョブの規模に応じて表 15 のキュー名を選択します．

表 15: kyu-cc のバッチキュー名

キュー名	CPU 時間	記憶容量	備考
sc	120 時間	4GB	非並列向け，省略キュー
sc8	120 時間	8GB	8CPU まで使用可
sc32	120 時間	32GB	32CPU まで使用可

○ sc キューに投入する例

ファイル “ a.sh ” に記述したバッチリクエストを qsub コマンドにより投入します．

```
kyu-cc% qsub a.sh ↵
Request 336.kyu-cc submitted to queue: sc.
```

上の例の “ 336 ” がリクエスト番号，“ kyu-cc ” がホスト名です．あわせた “ 336.kyu-cc ” を「バッチリクエスト識別子」と呼びます．バッチリクエスト識別子は，リクエストをキャンセルする時に必要となります．

○ sc8 キューに投入する例

-q に続けてキュー名を指定します．このオプションを省略した場合， sc キューに投入されます．

```
kyu-cc% qsub -q sc8 a.sh ↵
```

○ sc32 キューに投入する例

例では -eo オプションも併せて指定しています．qsub コマンドのオプションはバッチリクエストにも記述できます．

```
kyu-cc% qsub -q sc32 -eo a.sh ↵
```

○ qsub コマンドのオプション

よく用いる qsub コマンドのオプションを表 16 に示します．詳細は man qsub で参照してください．

表 16: よく用いる qsub コマンドのオプション

オプション	機能
-eo	標準エラー出力を標準出力ファイルへ出力します．通常は別々のファイルに出力されます．
-q *que-name*	*que-name* キューにジョブを投入します．省略した場合 sc キューに投入されます．
-lt *time*	CPU 時間のリミットを *time* に設定します．バッチキューの制限値以下が有効です．たとえば 1 時間 45 分に設定する場合は“ -lt 1:45:00 ”とします．
-me	実行終了をメールで通知します．
-o *filename*	標準出力を *filename* というファイルに出力します．

4.5 終了したバッチリクエストは

処理が終了すると，バッチリクエストファイル名とリクエスト番号に対応したファイルが返却されます．“ o ”のつく方に標準出力 (たとえば Fortran の装置番号 6 番)，“ e ”のつく方に標準エラー出力 (システムの発行するメッセージなど) が書き出されています．例えば，ファイル名“ a.sh ”を qsub コマンドによって投入し，リクエスト番号が“ 336 ”であった場合，標準出力ファイルは“ a.sh.o336 ”，標準エラー出力ファイルは“ a.sh.e336 ”という名前になります．バッチリクエストファイル名が長い場合は，適当な名前に変換される場合があります．

【注意】

標準出力ファイルの先頭には

```
Warning:  no access to tty; thus no job control in this shell...
```

また，ファイルの最後に“ logout ”という記述があります．この 2 つのメッセージは投入したバッチリクエストの処理結果とは**何の関係もありませんので，無視してください**．

4.6 バッチリクエストの状態表示

○ qstat コマンド

バッチキューの状態を表示するコマンドに qstat(/usr/bin/qstat) があります．バッチリクエスト識別子を確認する場合によく用います．

```
kyu-cc% qstat ↵          ←バッチリクエストの状態表示

sc@kyu-sc;  type=BATCH;  [ENABLED, INACTIVE];  pri=31
  0 exit;   5 run;   2 queued;   0 wait;   0 hold;   0 arrive;

sc8@kyu-sc;  type=BATCH;  [ENABLED, RUNNING];  pri=31
  0 exit;   1 run;   2 queued;   0 wait;   0 hold;   0 arrive;

         REQUEST NAME        REQUEST ID            USER  PRI    STATE      PGRP
    1:           a.sh       364.kyu-cc           a79999a  31  RUNNING     26034
                                ↑バッチリクエスト識別子の確認

sc32@kyu-sc;  type=BATCH;  [ENABLED, RUNNING];  pri=31
  0 exit;   4 run;   2 queued;   0 wait;   0 hold;   0 arrive;

         REQUEST NAME        REQUEST ID            USER  PRI    STATE      PGRP
    1:           a.sh       365.kyu-cc           a79999a  31  RUNNING     26049
    2:        test.sh       388.kyu-cc           a79999a  31   QUEUED
```

○ qps コマンド

投入したジョブの実行状況を調べるコマンドに qps(/usr/local/bin/qps) があります.
qps と入力すると，実行中のリクエストの状態を表示します.

```
kyu-cc% qps ↵                ←リクエストの状態表示
    UID    PID   PPID    STIME    REQ-ID  TIME COMMAND
a79999a  26051  26050  14:29:27     365   0:00 -csh
a79999a  26087  26052  14:29:34     365   0:00 timex ./a.out
a79999a  26052  26051  14:29:27     365   0:00 /bin/csh /usr/spool/nqs/scripts/6048
a79999a  26088  26087  14:29:34     365  13:28 ./a.out          ← 13 分 28 秒実行中
```

4.7　バッチリクエストのキャンセル

何らかの理由で投入したバッチリクエストをキャンセルするには，qdel(/usr/bin/qdel) コマンドを用います．キャンセルのためには，qstat コマンドによってバッチリクエスト識別子を確認しておく必要があります.

○実行待ちのバッチリクエストをキャンセル

qstat コマンドによって表示される STATE が QUEUED のバッチリクエストが対象です．qdel に続けてバッチリクエスト識別子を指定します.

```
kyu-cc% qdel 338.kyu-cc ↵        ←実行待ちのリクエストをキャンセル
Request 338.kyu-cc has been deleted.
```

○実行中のバッチリクエストをキャンセル

qstat コマンドによって表示される STATE が RUNNING のバッチリクエストが対象です．-k オプションを指定します.

```
kyu-cc% qdel -k 338.kyu-cc ↵        ←実行中のリクエストをキャンセル
Request 338.kyu-cc is running, and has been signalled.
```

4.8 VPP700/56 に投入するバッチリクエスト

kyu-cc からスーパーコンピュータ VPP700/56(ホスト名: kyu-vpp) にジョブを投入することができます． ただし，ソースプログラム，データファイル，バッチリクエストファイルなどは必ずホームディレクトリにある “ VPP ” という名前のディレクトリにコピーまたは移動してください．VPP ディレクトリが kyu-vpp のホームディレクトリとなります．

kyu-vpp と kyu-cc では，コマンド，オプションが異なる場合があります．また， kyu-cc で作成した実行可能ファイルを kyu-vpp で実行することはできません．詳しい利用方法は [25] を参照してください．

4.8.1 バッチリクエストの投入

kyu-cc から kyu-vpp へのバッチリクエストの投入は qsub コマンドによって行ないます．投入するジョブの規模に応じて表 17 のキュー名を選択します． kyu-vpp のキュー名を選択すると，ジョブは自動的に kyu-vpp で実行されます．

表 16: kyu-vpp のバッチキュー名

キュー名	CPU 時間	記憶容量		処理形態
		最大値	省略値	
c	60 分	0.5GB		翻訳専用
s	60 分	1.7GB	0.5GB	1PE
p1	1200 分	1.7GB	0.5GB	1PE
s8	10 分	1.7GB/PE	0.5GB/PE	最大 8PE
p8	1200 分	1.7GB/PE	0.5GB/PE	最大 8PE
p16	1200 分	1.7GB/PE	0.5GB/PE	最大 16PE
p32	1200 分	1.7GB/PE	0.5GB/PE	最大 32PE

○ kyu-vpp の p1 キューに投入する例

kyu-cc のホームディレクトリの “ VPP ” が kyu-vpp のホームディレクトリ，すなわち kyu-vpp に投入するバッチリクエストの始まりのディレクトリです．あらかじめ VPP ディレクトリ以下にファイルをコピーしてください．以下は， kyu-vpp の p1 キューに投入する例です．

```
kyu-cc% qsub -q p1 a.sh ↵
```

4.8.2 バッチリクエストの状態表示

qstat，qps コマンドに @kyu-vpp オプションを指定します．

```
kyu-cc% qstat @kyu-vpp ↵    ← kyu-vpp のバッチリクエストの状態表示
```

```
kyu-cc% qps @kyu-vpp ↵      ← kyu-vpp の実行状況を調べる
```

4.8.3 バッチリクエストのキャンセル

○実行待ちのバッチリクエストをキャンセル

qdel コマンドに -r kyu-vpp オプションを指定します.

```
kyu-cc% qdel -r kyu-vpp 3519.kyu-cc ↵        ←実行待ちのリクエストをキャンセル
Request 3519.kyu-cc has been deleted.
```

○実行中のバッチリクエストをキャンセル

-r kyu-vpp -k オプションを指定します.

```
kyu-cc% qdel -r kyu-vpp -k 3519.kyu-cc ↵        ←実行中のリクエストをキャンセル
Request 3519.kyu-cc is running, and has been signalled.
```

おわりに

本稿では汎用 UNIX サーバにおけるプログラミング言語の基本的な利用方法について説明しました. 今後の機能追加などにより利用方法が変更になる可能性もあります. 最新の情報はセンターホームページ (http://www.cc.kyushu-u.ac.jp/) 上で適宜公開していく予定です.

汎用 UNIX サーバの利用方法に限らず, センターに関する質問・要望は

request@cc.kyushu-u.ac.jp

で受け付けています.

参考文献

[1] FUJITSU Fortran 使用手引書 V4 用, 富士通株式会社, J2X0-3350, 1998.

[2] FUJITSU Fortran 文法書, 富士通株式会社, 1999. (PDF 形式; 376 ページ; 5.4MB)

[3] Fortran 翻訳時メッセージ, 富士通株式会社, 1999. (HTML 形式)

[4] Fortran 実行時メッセージ, 富士通株式会社, 1999. (HTML 形式)

[1]–[4] はhttp://www.cc.kyushu-u.ac.jp/GP7000F_MANUAL/japanese/Fortran/index.html を参照

[5] FUJITSU C 言語使用手引書 V4 用, 富士通株式会社, J2X0-3340, 1998.
http://www.cc.kyushu-u.ac.jp/GP7000F_MANUAL/japanese/C/index.htm

[6] FUJITSU C++ 言語使用手引書 V4 用, 富士通株式会社, J2X0-0260, 1998.
http://www.cc.kyushu-u.ac.jp/GP7000F_MANUAL/japanese/C++/index.htm

[7] 富士通 SSL II 使用手引書 (科学用サブルーチンライブラリ), 富士通株式会社, 99SP-4020, 1987.

[8] FUJITSU SSL II 拡張機能使用手引書 (科学用サブルーチンライブラリ), 富士通株式会社, 99SP-4070, 1991.

[9] 📖 FUJITSU SSL II 拡張機能使用手引書 II(科学用サブルーチンライブラリ) V4.0 用, 富士通株式会社, J2X0-1366, 1997.

[7]–[9] はhttp://www.cc.kyushu-u.ac.jp/library/SSL2/SSL2.html から PDF ファイルも参照可能

[10] 📖 FUJITSU C-SSL II 使用手引書 (科学用関数ライブラリ), 富士通株式会社, J2X0-3330, 1997.
http://www.cc.kyushu-u.ac.jp/library/SSL2/C-SSL2.html

[11] 📖 NUMPAC 利用手引書, 富士通株式会社, 1994.
http://cronos.fuis.fukui-u.ac.jp/numpac/

[12] BLAS, LAPACK, ScaLAPACK 使用手引書, 富士通株式会社, 1999. (PDF 形式; 33 ページ; 133KB)
http://www.cc.kyushu-u.ac.jp/library/ScaLAPACK/ScaLAPACK.html

[13] http://www.cc.kyushu-u.ac.jp/library/NAG/NAG.html

[14] http://www.cc.kyushu-u.ac.jp/library/IMSL/IMSL.html

[15] http://www.cc.kyushu-u.ac.jp/library/CalcPS/CalcPS.html

[16] Fortran/MP 使用手引書 V1 用, 富士通株式会社, 1999.(HTML 形式)
http://www.cc.kyushu-u.ac.jp/library/OpenMP/OpenMP.html

[17] MPI 使用手引書 V2.1 用, 富士通株式会社, 1999.(HTML 形式)
http://www.cc.kyushu-u.ac.jp/library/MPL/MPI.html

[18] Bradford Nichols, Dick Buttlar, Jacqueline Proulx Farrell (榊 正憲 訳): Pthreads プログラミング, オライリー・ジャパン, 1998.

[19] Bil Lewis, Daniel J. Berg (岩本 信一 訳): マルチスレッドプログラミング入門, アスキー出版社, 1996.

[20] 池田 大輔: UNIX での遠隔接続とファイル転送, 九州大学大型計算機センター広報, Vol.33, No.1 (本号).

[21] 伊東 栄典: Windows95/98/NT での遠隔接続とファイル転送, 九州大学大型計算機センター広報, Vol.33, No.1 (本号).

[22] 伊東 栄典, 平野 広幸: MAC OS での遠隔地接続とファイル転送, 九州大学大型計算機センター広報, Vol.33, No.1 (本号).

[23] 南里 豪志: 大型計算機センターの UNIX 入門, 九州大学大型計算機センター広報, Vol.31, No.2, pp.61-102 (1998).

[24] 汎用エディター je(β 版) 使用手引, 日本原子力研究所情報システム管理課, 1998 (Microsoft Word 文書; 969KB).
http://www.cc.kyushu-u.ac.jp/library/je/je.html

[25] 📖 VPP700/56 利用の手引 (第 2.0 版), 九州大学大型計算機センター, 1998 年 6 月. 184 ページ.
http://www.cc.kyushu-u.ac.jp/RD/watanabe/RESERCH/MANUSCRIPT/MANUAL/VPP700/intro.html

📖 の印のあるものは, 冊子版をセンター図書室で参照することができます.