VPP5000/64 利用法

渡部善隆 *, 田中省作 *

本稿では , 2001 年 1 月から運用を開始した新スーパーコンピュータ FUJITSU VPP5000/64(ホスト名: kyu-vpp) の利用方法を , Fortran, C, C++ プログラムの翻訳・実行方法を中心に説明します .

1 VPP5000/64 への接続

1.1 ホスト名

ネットワーク経由で接続する場合のホスト名は以下の通りです .

ホスト名	kyu-vpp
ドメイン名	cc.kyushu-u.ac.jp
IP アドレス	133.5.8.160

telnet, rlogin などで kyu-vpp に接続する場合には ,「ホスト名+ドメイン名」の
" kyu-vpp.cc.kyushu-u.ac.jp" または IP アドレスの" 133.5.8.160" を指定します . IP アドレスは予告なしに変更になる場合がありますので ,「ホスト名+ドメイン名」の指定方法をお勧めします .

! VPP700/56 のときとは IP アドレスが変更になっています .

詳細は以下のページを参照してください.

接続方法について

- UNIX からの遠隔ログイン
  http://www.cc.kyushu-u.ac.jp/scp/system/manual/etc/RemoteLogin.html
- MS-Windows での Telnet と Secure Shell(SSH) の利用方法
  http://www.cc.kyushu-u.ac.jp/scp/system/general/GP7000F/telnet.win/
- Macintosh での Telnet 利用方法
  http://www.cc.kyushu-u.ac.jp/scp/system/general/GP7000F/telnet.mac/telnet.html

ファイルの転送方法について

- UNIX からのファイル転送
  http://www.cc.kyushu-u.ac.jp/scp/system/manual/etc/FTP.html

*情報基盤センター 研究部 E-mail: {watanabe, sho}@cc.kyushu-u.ac.jp

- WS_FTP の使い方
  http://www.cc.kyushu-u.ac.jp/scp/network/WSFTP/index-j.shtml
- MacOS でのファイル転送
  http://www.cc.kyushu-u.ac.jp/scp/system/general/GP7000F/telnet.mac/ftp.html

1.2 オペレーティングシステム

kyu-vpp のオペレーティングシステム (OS) は，UNIX OS の一種である **UXP/V V20** です．UNIX を使ったことのある方ならば，違和感なく kyu-vpp 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 [4] を参照してください．

1.3 kyu-cc からのファイル参照

汎用 UNIX サーバ kyu-cc からスーパーコンピュータ VPP5000/64 のファイルを 参照することができます．kyu-cc ホームディレクトリ (ログイン直後の利用者ディレクトリ) の" VPP"という名前のディレクトリと 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．ディレクトリ VPP に移動することにより，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に指定したり，kyu-vpp のファイルの編集，バッチジョブの投入などを行うことができます．

1.4 エディタ

kyu-vpp で利用できるエディタは以下の通りです．

エディタ名	コマンド名	備考
emacs	emacs(/usr/local/bin/emacs)	GNU Emacs 20.7.1; 日本語入力は Free Wnn 1.1
vi	vi(/usr/bin/vi)	利用方法は man vi で参照して下さい
je	je(/usr/local/bin/je)	PFD[1]ライクなエディタ

1.5 VPP5000/64 用環境設定ファイル

2000 年末まで稼動していた旧スーパーコンピュータ VPP700/56 から継続して VPP5000/64 を利用されている場合には，.cshrc，.login，.profile ファイルは VPP5000/64 に コピーされています．センターでは VPP5000/64 用のファイルの雛型も以下に用意していますので，これを機会に変更することをお勧めします．

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

[1] 旧汎用計算機 (OS は MSP) で使用されていたエディタです．

コピー方法は以下の手順を参考にしてください．

```
kyu-vpp% mv .cshrc .cshrc.vpp700                    ←バックアップファイルを作成
kyu-vpp% cp /usr/local/skel/local.cshrc .cshrc      ← VPP5000/64 用 .cshrc ファイル
                                                      をコピー
```

以上の操作では，現在使用中の csh に，新しい .cshrc の設定は反映されません．変更を直ちに反映させるには，次のように source コマンドを実行する必要があります．

```
kyu-vpp% source .cshrc
```

2 ソフトウェア一覧

VPP5000/64 で利用できるプログラム言語，数値計算ライブラリ，アプリケーションソフトウェアの一覧を示します．

2.1 プログラム言語

ソフトウェア名	コマンド	機能
Fortran	frt	JIS X 3000-1 完全準拠 Fortran 95 コンパイラ
VPP Fortran	frt -Wx	ベクトル並列化コンパイラ
HPF	frt -Wh	分散メモリ型並列計算機向け言語
C	cc -Kvp	ANSI 規格準拠，K&R 仕様 [2] をサポートした C コンパイラ
C++	CC	ARM 仕様 [1] 準拠の C++コンパイラ
DPCE	dpcc	データ並列処理用 C コンパイラ

2.2 プログラム言語開発支援

ソフトウェア名	機能
Analyzer	チューニングおよびデバッグ支援ツール
Workbench	GUI 環境によるプログラム開発支援ツール
MPTools	メッセージパッシングプログラム解析ツール
TotalView	メッセージパッシングプログラムデバッグ支援ツール

2.3 メッセージパッシングライブラリ

ソフトウェア名	機能
MPI	MPI 2.0 規格準拠のメッセージパッシングライブラリ
PVM	PVM 3.3 準拠のメッセージパッシングライブラリ

2.4 数値計算ライブラリ

ソフトウェア名	機能
SSL II/VP	Fortran 用汎用数値計算ライブラリ
SSL II/VPP	SSL II/VP の並列版
C-SSL II/VP	SSL II/VP サブルーチンとの C 言語 , C++言語インターフェース
NUMPAC	Fortran 用数値計算パッケージ
LAPACK/VP, BLAS/VP	線形計算ライブラリ
ScaLAPACK	並列版線形計算ライブラリ
IMSL Fortran 90 MP Library	MPI 環境の並列ライブラリ (Fortran 90 用)
IMSL C Library	C 用関数ライブラリ
SALS	最小二乗法プログラムパッケージ

2.5 計算化学

ソフトウェア名	機能
Gaussian 98	非経験的分子軌道計算プログラム
MOPAC2000	半経験的分子軌道計算プログラム
xmo	Gaussian, MOPAC2000 用可視化プログラム
MASPHYC / MASPHYC-SP	材料の物性・構造予測システム

2.6 アプリケーション・ソフトウェア

ソフトウェア名	機能
α-FLOW	汎用 3 次元流体解析システム
Marc / Mentat	汎用有限要素法解析プログラム
Nastran / Patran	汎用非線形構造解析プログラム
LS-DYNA	非線形動的構造解析プログラム

2.7 プログラムライブラリ開発課題

Fortran サブルーチンライブラリ

プログラム名	機能
CGJQ Gauss-Jacobi	積分公式の分点，重み算定
CGLQ Gauss-Laguerre	積分公式の分点，重み算定
DIFF1S, DIFF1D	解析関数の数値微分 (単・倍精度)
MINMAX	線形方程式のミニマックス解
PRESNL	一般化されたフレネル積分
CA01	計量修正による関数の極小点発見
DA02	BCS 方程式
DB01, DB02, DB03	Clebsch-Gordan, Racah, 9-J 係数
KNL1	直交条件模型の積分核
VAR1	微積分方程式の散乱境界条件解
AACOUST	建築音響解析ライブラリ

詳細は，次のページを参照してください．

http://www.cc.kyushu-u.ac.jp/scp/system/library/PROGRAM_LIBRARIES/Fortran_Subroutines/

3 利用形態と制限値

3.1 利用形態

kyu-vpp でプログラムを翻訳・編集結合・実行する方法は，大きく対話型処理とバッチ処理に分かれます．対話型処理とは，コマンドを入力することによって翻訳処理や実行などをインタラクティブに行う利用方法のことです．対話型処理は，翻訳やデバッグ，小規模なプログラムの実行に適しています．バッチ処理とは，一連の処理を記述した「バッチリクエスト」と呼ばれるシェルスクリプトによって処理を行う方法です． バッチ処理は，多くの記憶領域を必要とするプログラムや並列度の高いジョブに適しています．

なお，UNIX でよく行う，コマンドラインの最後に" &"をつける処理方法 (バックグラウンドジョブ) 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．

3.2 制限値

対話型処理，バッチ処理の制限値は以下の通りです．バッチ処理の詳細は「7. バッチ処理」を参照してください．

処理形態	キュー名	記憶容量		演算時間	備考
		省略値	制限値		
対話型処理	–	1GB		1 時間	非並列
バッチ処理	s	2GB	7GB	1 時間	非並列・短時間向き
	p1	2GB	14.5GB	20 時間	非並列 (省略キュー)
	s8	2GB/PE	7GB/PE	1 時間	8PE まで使用可・短時間向き
	p8	2GB/PE	7GB/PE	20 時間	8PE まで使用可
	p16	2GB/PE	7GB/PE	20 時間	16PE まで使用可
	p32	2GB/PE	7GB/PE	20 時間	32PE まで使用可
	x8	10GB/PE	15GB/PE	20 時間	8PE まで使用可・大規模記憶容量向き
	x16	10GB/PE	15GB/PE	20 時間	16PE まで使用可・大規模記憶容量向き

4 Fortran の対話型処理

より詳しい利用方法は次のオンラインマニュアル [3] を参照してください．

- UXP/V Fortran 使用手引書 V20 用
- UXP/V Fortran メッセージ説明書 V20 用
- UXP/V Fortran プログラミングハンドブック V20 用
- UXP/V Fortran User's Guide V20
- UXP/V Fortran Messages (V20)
- UXP/V Fortran Programming Handbook V20

4.1 コマンドとファイル拡張子名

Fortran の翻訳と結合編集のコマンドは frt です．kyu-vpp にログイン後，man frt と入力すると，機能の詳細を表示させることができます．Fortran ソースファイルの拡張子はプログラムの形式に応じて以下のようにしてください．

拡張子	プログラムの種類
.f90 または .f95	自由形式
.f	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．

! 拡張子 ".f90" または ".f95" のファイルは通常「自由形式で記述された Fortran 90 または Fortran 95 プログラム」と解釈されます．これに対し，拡張子 ".f" のファイルは通常「固定形式で記述された Fortran 77 プログラム」と解釈されます．したがって，拡張子 ".f" のファイルに Fortran 90 または Fortrn 95 から新たにサポートされた機能を記述する場合には，翻訳時オプション-X9 が必要になります．

4.2 基本的な手順

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

処理が致命的なエラーを起こすことなく終了した場合，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-vpp% ./a.out    ←実行
```

! 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-vpp(VPP700/56) では“ a.out” と指定するだけでプログラムを実行することができました． しかし，セキュリティ強化のため，新 kyu-vpp ではデフォルトではカレントディレクトリのパスを設定していません．必ず最初に“ ./” を指定してください．

4.3 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．

```
kyu-vpp% 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”，副プログラム群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-vpp% frt -c sub.f    ←オブジェクトファイルの作成（プログラムは固定形式）
```

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します．

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

このようにすると，副プログラムの翻訳は一度だけで済むため，メインプログラムを何度も修正する場合に全体の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も指定することができます．また，複数のソースファイル，オブジェクトファイルも指定できます．

4.4 よく用いる翻訳時オプション

よく用いる Fortran の翻訳時オプションを以下に示します．詳細はオンラインマニュアル [3] を参照してください．

-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．
-Dasux	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	モジュールを翻訳します．
-Ob	最適化を基本的な機能のみに制限します．
-O5	最大限の最適化を試みます．数学的に等しい範囲で括弧を無視した変更等を行うため，場合によっては 計算結果に大きな差が生じることがあります．指定する場合には注意が必要です．
-Kfast	VPP5000/64 に適した最適オプションを自動選択 (コンパイラに” おまかせ”) します．
-Eeipu	最適化に関するメッセージを出力します．
-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 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．
-Pa	ソースプログラムと対比させながら配列演算に対するベクトル化情報を出力します．
-Z *filename*	翻訳に関する情報を *filename* に書き出します．
-KA32	オブジェクトファイルを 32 ビットアドレスモードで作成します．省略値は 64 ビットモードです．プログラムによっては実行性能が向上する場合があります．ただし，64 ビットアドレスモードのオブジェクトファイルとの混在はできない上に，記憶容量の最大値が 2GB に制限されるため，指定する場合には注意が必要です．

以下は，デバッグオプションを指定して翻訳・編集結合し，実行する例です．

```
kyu-vpp% frt -Dasux example.f95          ←デバッグオプションの指定
kyu-vpp% ./a.out                         ←実行を通してデバッグ
 jwe0340i-e line 7 ベクトル化された配列要素または文字部分列 ( M ) の引用で，
2 次元目の添字式または部分列式の値 ( 1002 ) は，宣言した範囲 ( 1:1000 ) 内で
なければなりません．
      :
```

以下は”おまかせ”最適化オプションの指定 (-Kfast) とともに，プログラムリストと対比させたベクトル化情報 (-Pa) と最適化情報 (-Eeipu) を出力させ，結果をファイル” clist”に書き出す例です．

```
kyu-vpp% frt -Kfast -Pa -Eeipu -Z clist example.f90     ←オプションの指定例
```

この場合，オプションの順番は任意です．

4.5 よく用いる実行時オプション

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以下に示します．詳細はオンラインマニュアル [3] を参照してください．

-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 の指定により行われる IBM370-IEEE 形式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．
-Oi	入出力に関する統計情報を出力します．

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い，あわせて診断メッセージを出力する例です．

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

4.6 最適化オプション

Fortran の最適化レベルは標準レベルである" -Oe"(-O3 と等価) です．" -Oe"レベルでは，演算評価方法の変更，不変式の先行評価，総和演算のベクトル化などが行われます．これらの最適化の副作用として，計算結果に違いが生じたり，正しいプログラムにもかかわらずごくまれに実行時に異常終了したりすることがあります．最適化機能の使用上の注意についてはオンラインマニュアル [3] を参照してください．

4.6.1 基本最適化レベルでの翻訳

最適化を基本レベルに抑えた翻訳を行う場合には，" -Ob"オプションまたは " -O2"オプションを指定します．作成したプログラムの精度を標準オプションと確認する場合などにも利用できます．

```
kyu-vpp% frt -Ob main.f90     ←最適化オプションを下げる
kyu-vpp% ./a.out              ←実行
```

4.6.2 最適化オプションの指定例

-Kfast オプションを指定して" おまかせ最適化 "をおこないます．翻訳時間は一般に増大します．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．

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

4.6.3 最大限の最適化オプション

最大限の最適化オプションは-O5 です．さらに，記憶領域が 2GB 以下であれば，-KA32 オプションを指定することで，さらに高速になる場合があります．ただし，実行結果に副作用が生じる可能性があり，-KA32 オプションを指定して作成したオブジェクトファイルは通常の 64 ビットアドレスモードのオブジェクトファイルと互換性がありません．使用する場合は注意してください．

```
kyu-vpp% frt -O5 -KA32 main.f90          ←最大限の最適化
```

4.7 時間計測コマンド

timex(/usr/bin/timex) コマンドにより，翻訳，実行などの経過時間，CPU 時間を計測することができます．

```
kyu-vpp% timex frt example.f95      ←翻訳と結合編集の時間計測
real             4.01               ←経過時間 (4 秒 1)
user             1.70               ←ユーザ CPU 時間
sys              0.80               ←システム CPU 時間
vu-user          0.00
vu-sys           0.00

kyu-vpp% timex ./a.out              ←実行時間の計測
eal             10.30               ←経過時間 (10 秒 3)
user            10.24               ←ユーザ CPU 時間
sys              0.03               ←システム CPU 時間
vu-user          8.21               ←ユーザ CPU 時間の中でベクトルユニットが
                                      動作した時間
vu-sys           0.00               ←システム CPU 時間の中でベクトルユニット
                                      が動作した時間
```

4.8 数値計算ライブラリの組み込み

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 コマンドのオプション-l を用いてライブラリを指定します．-l は結合編集時のオプションのため，通常は次のように，コマンド並びの最後に指定してください．

```
kyu-vpp% frt main.f90 -lssl2vp       ← SSL II/VP の組み込み
```

利用できるライブラリとオプションは以下の通りです．

ライブラリ名	オプション
SSL II/VP	-lssl2vp
SSL II/VPP	-lssl2vpp (実行可能ファイルを作成するまで)
NUMPAC	-lnumpac
BLAS	-lblasvp
LAPACK	-llapackvp -lblasvp
ScaLAPACK	-scalapack
IMSL Fortran 90 MP Library	http://www.cc.kyushu-u.ac.jp/scp/system/library/IMSL/kyu-vpp.html を参照してください
プログラムライブラリ開発課題分	-lsslq

4.9 ファイル入出力

ここでは，OPEN 文にファイル名を陽に指定しない場合のファイル入出力について簡単に紹介します．詳細はオンラインマニュアル [3] を参照してください．

4.9.1 標準入出力

装置番号 5 番 (READ(5) に対応) と 6 番 (WRITE(6)) 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．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-vpp% ./a.out < in.data                ←装置番号 5 番から in.data の内容を読み込む
kyu-vpp% ./a.out > out.data               ←装置番号 6 番への出力結果を out.data に保存
kyu-vpp% ./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-vpp% ./a.out >& out.data              ←装置番号 6 番への出力結果および実行時
                                            のエラーメッセージを out.data に保存
kyu-vpp% ./a.out < in.data > out.data     ← in.data の内容を読み込み out.data に保存
```

4.9.2 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“ fuXX”で行います．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力を行った場合は“ fort.XX”というファイルに出力されます．

以下の例は，装置番号 10 番とファイル“ example.data”を結合し，実行する例です．

```
kyu-vpp% setenv fu10 example.data          ←ファイルと装置番号の結合
kyu-vpp% ./a.out                           ←実行
```

4.9.3 書式なし入出力

書式なし入出力を行う場合には，OPEN 文に FORM='UNFORMATTED' を指定してください．Fortran 95 仕様からは，明示的に指定しないとエラーとなります．

```
REAL(KIND=8),DIMENSION(100) :: X                ←配列の宣言
  :
OPEN(1,FILE='EXAMPLE.DATA',FORM='UNFORMATTED') ←書式なしデータを開く
READ(1) X                                       ←書式なし入力
CLOSE(1)                                        ←ファイルを閉じる
```

5 C, C++ の対話型処理

より詳しい利用方法は次のオンラインマニュアル [3] を参照してください．

- UXP/V C 言語使用手引書 V20 用
- UXP/V C++オンラインマニュアル
- UXP/V C Language User's Guide V20
- UXP/V C++ Online Manual

5.1 コマンドとファイル拡張子名

C の翻訳と結合編集のコマンドは cc です．C++の翻訳と結合編集のコマンドは CC です．cc コマンドに" -Kvp"オプションを指定することにより，ベクトル翻訳を行います．省略した場合にはスカラー翻訳を行います．CC コマンドでは" -Kvp"オプションが省略値です．

> ! VPP700/56 では C のスカラー翻訳コマンドが cc，ベクトル翻訳コマンドが vcc でした．VPP5000/64 では，cc コマンドのオプションで使い分けるように変更になっています．

ソースファイルの拡張子は以下の名前にしてください．

プログラムの種類	拡張子
C	.c
C++	.cc または.C または.c

5.2 基本的な手順

cc コマンドまたは CC 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-vpp% cc 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・スカラー翻訳)
```

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-vpp% ./a.out    ←実行
```

ベクトル翻訳を行う場合には“ -Kvp ”オプションを指定します．配列要素の繰返し演算が多く含まれているプログラムの場合，実行時間が大幅に短縮することが期待されます．

```
kyu-vpp% cc -Kvp 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・ベクトル翻訳）
```

5.3 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．

```
kyu-vpp% cc -c example.c    ←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.c ”，副プログラム群を記述したファイルを“ sub.c ”とします．まず，sub.c 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-vpp% cc -c sub.c    ←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します．

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

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

cc, CC コマンドには上の例のようにオブジェクトファイルも指定することができます．また，複数のソースファイル，オブジェクトファイルも指定できます．

5.4 よく用いる翻訳時オプション

よく用いる C, C++の翻訳時オプションを以下に示します．詳細はオンラインマニュアル [3] を参照してください．

-Kvp	ベクトル翻訳を行うことを指示します．
-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．
-O -K4	最大限の最適化を指示します．副作用が生じる可能性がありますので，指定する場合には注意が必要です．
-Wv,-m3	ベクトル化メッセージの情報を出力します．
-Ksrc	ソースリストに対応した最適化・ベクトル化情報を出力します．
-Ksta	統計情報を出力します．
-Z *filename*	翻訳に関する情報を *filename* に書き出します．長大なエラーメッセージや-Ksrc オプションの結果をファイルに書き出す場合に利用します．
-KA32	オブジェクトファイルを 32 ビットアドレスモードで作成します．省略値は 64 ビットモードです． プログラムによっては実行性能が向上する場合があります．ただし，64 ビットアドレスモードのオブジェクトファイルとの混在はできない上に，記憶容量の最大値が 2GB に制限されるため，指定する場合には注意が必要です．

以下は，最適化オプションを指定してベクトル翻訳・編集結合し，実行する例です．

```
kyu-vpp% cc -Kvp -O -K4 example.c    ←最適化オプションの指定
kyu-vpp% ./a.out                     ←実行
```

5.5 数値計算ライブラリの組み込み

ライブラリ名	オプション
C-SSL II/VP	-Wg, -S -DMAIN_=main
IMSL C Library	($CFLAGS) と ($LINK_CNL) の設定

数値計算ライブラリの利用方法は以下のページを参考にしてください．

- C-SSL II/VP
  http://www.cc.kyushu-u.ac.jp/scp/system/library/SSL2/C-SSL2.html
- IMSL C Library
  http://www.cc.kyushu-u.ac.jp/scp/system/library/IMSL/IMSL.html

6 並列プログラムの対話型翻訳

VPP5000/64 は分散メモリ型ベクトル並列計算機です．現在のところ，自動並列化機能は有していません．複数 PE を利用した並列計算のためには，利用者自身で並列化を行う必要があります．

VPP5000/64 で可能な並列化手法は以下の通りです．

ソフトウェア名	並列化の概要
VPP Fortran	Fortran プログラムに指示行を挿入
HPF	Fortran プログラムに指示行を挿入
DPCE	ANSI-C 言語のデータパラレル型拡張仕様
MPI	Fortran, C プログラムからライブラリ関数を呼出す
PVM	Fortran, C プログラムからライブラリ関数を呼出す

VPP5000/64 の並列プログラムは，対話型に翻訳，結合・編集を行い，実行可能ファイルを作成することができます．実行はバッチ処理で行います．以下，各並列化の概要を説明します．

6.1 VPP Fortran

詳細は次のオンラインマニュアル [3] を参照してください．

- UXP/V Fortran/VPP 使用手引書 V20 用
- UXP/V VPP Fortran プログラミングハンドブック V20 用
- UXP/V Fortran/VPP User's Guide (V20)
- UXP/V VPP Fortran Programming Handbook (V20)

VPP Fortran プログラムの翻訳のためには，frt コマンドに-Wx オプションを指定します．このオプションの指定によって，VPP Fortran コンパイラが起動され， ソースプログラムに挿入した「拡張最適化制御行」が有効になります．

```
kyu-vpp% frt -Wx example.f90      ← VPP Fortran コンパイラの起動
```

6.2 HPF(High Performance Fortran)

詳細は次のオンラインマニュアル [3] を参照してください．

- UXP/V HPF 使用手引書 V20L20 L00061 用
- UXP/V HPF User's Guide (V20L20 L00061)

HPF プログラムの翻訳のためには，frt コマンドに-Wh オプションを指定します．このオプションの指定によって，HPF コンパイラが起動され，ソースプログラムに挿入した「HPF 指示行」が有効になります．

```
kyu-vpp% frt -Wh example.f90      ← HPF コンパイラの起動
```

6.3 DPCE(Data-Parallel C Extensions)

詳細は次のオンラインマニュアル [3] を参照してください．

- UXP/V DPCE 使用手引書 V20L20

- UXP/V DPCE User's Guide V20

DPCE プログラムの翻訳および実行可能ファイルの作成は kyu-vpp の dpcc コマンドによって行います．DPCE ソースプログラムの拡張子は" .dpc "とします．

```
kyu-vpp% dpce example.dpc      ← DPCE プログラムの翻訳，編集結合
```

6.4 MPI(Message Passing Interface)

詳細は次のオンラインマニュアル [3] を参照してください．

- UXP/V MPI 使用手引書 V20 用
- UXP/V MPI User's Guide V20

MPI プログラムの翻訳および実行可能ファイルの作成は kyu-vpp の mpifrt コマンド (Fortran の場合) または mpicc コマンド (C の場合) によって行います．

```
kyu-vpp% mpicc example.c      ← MPI プログラムの翻訳，編集結合
```

6.5 PVM(Parallel Virtual Machine)

詳細は次のオンラインマニュアル [3] を参照してください．

- UXP/V PVM 使用手引書 V20 用
- UXP/V PVM User's Guide V20

PVM プログラムの翻訳および実行可能ファイルの作成は kyu-vpp の frt コマンド (Fortran の場合) または cc コマンド (C の場合) に PVM ライブラリを指定して行います．以下は，指定例です[2]．詳細はマニュアルを参照してください．

```
kyu-vpp% cc  test.c -Wl,-P \
? -L/usr/lang/pvm64/lib/UXPV \
? -J -dy -lpvm -lmp -lgen -lelf -lsocket -lpx \
?  -lc -I/usr/lang/pvm64(pvm32)/include            ← PVM プログラムの翻訳，編集結合
```

" "

7 バッチ処理

対話型の制限値を超えるジョブおよび並列プログラムの実行はバッチ処理で行います．

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．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以下の通りです．

[2]指定するオプションが多い場合など，例のように" \" を入れることで，複数行に渡ってコマンドを入力することができます．

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

7.1 バッチリクエストの書き方

バッチリクエストはファイルとしてエディタにより作成します．ファイル名は任意です．ここでは“ a.sh”という名前とします．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
frt 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．この記事では csh の記述にしたがっています．よくわからない方は「おまじない」だと思って記述してください．また，mule, emacs で拡張子“ .sh”を付けて新規にファイルを作成する場合，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 (ログイン時のディレクトリ) 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のコマンドと同様です．ここでは“ main.f90”を翻訳・結合編集し，実行しています．

7.1.1 バッチリクエストの記述例

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
```

2. Fortran プログラムを翻訳・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
frt main.f90
./a.out < in.data
```

3. Fortran プログラムを翻訳・結合編集し，実行します．編集結合時に SSL II/VP 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ってください．

```
#
cd mydir
frt main.f90 -lssl2vp
setenv fu01 inout1.data
setenv fu23 inout2.data
./a.out
```

4. 最適化オプション-O5 を指定して 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 frt -O5 main.f90
timex ./a.out
```

5. VPP Fortran プログラムを翻訳・結合編集し，実行します．翻訳時オプション-Wx が必要です．

```
#
cd mydir
frt -Wx main.f90
./a.out
```

6. C プログラム“ main.c”をベクトル翻訳・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
cc -Kvp main.c
./a.out < in.data > out.data
```

7. C++プログラム“ main.C”をベクトル翻訳・結合編集し，実行します．

```
#
cd mydir
CC -Kvp main.C
./a.out
```

8.“ #@$”に続けて，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，p8 キューに投入することを指示する-q p8 オプションを記述しています．また，VPP Fortran プログラムを翻訳するオプション“ -Wx”を指定し，実行可能ファイル名を“ b.out”に変更しています．

```
#
#@$-q p8
cd mydir
frt -Wx -o b.out main.f90
./b.out
```

9. あらかじめ作業を行うディレクトリに" 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・編集結合し，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指定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
frt main.f95
./a.out
```

7.2 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います．投入するジョブの規模に応じて以下のキュー名を選択します．

キュー名	記憶容量		演算時間	備考
	省略値	制限値		
s	2GB	7GB	1 時間	非並列・短時間向き
p1	2GB	14.5GB	20 時間	非並列 (省略キュー)
s8	2GB/PE	7GB/PE	1 時間	8PE まで使用可・短時間向き
p8	2GB/PE	7GB/PE	20 時間	8PE まで使用可
p16	2GB/PE	7GB/PE	20 時間	16PE まで使用可
p32	2GB/PE	7GB/PE	20 時間	32PE まで使用可
x8	10GB/PE	15GB/PE	20 時間	8PE まで使用可・大規模記憶容量向き
x16	10GB/PE	15GB/PE	20 時間	16PE まで使用可・大規模記憶容量向き

7.2.1 p1 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-vpp% qsub a.sh
Request 211.kyu-vpp submitted to queue: p1.
```

この例の" 211" がリクエスト番号", kyu-vpp" がホスト名です．これらをあわせた" 211.kyu-vpp " 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，特にリクエストをキャンセルする時に必要となります．

7.2.2 p8 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，p1 キューに投入されます．

```
kyu-vpp% qsub -q p8 a.sh
```

7.2.3 p16 キューに投入する例

以下の例ではエラーメッセージもファイルに出力させるため-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-vpp% qsub -q p16 -eo a.sh
```

7.2.4 よく用いるqsub のオプション

詳細は man qsub で参照してください．

-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 p1 キューに投入されます．
-lM *size*	*size* まで記憶容量を使用することを指定します．投入するバッチキューの制限値以下の値だけが有効です．たとえば 10GB に設定する場合は" -lM 10gb" とします．
-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の値だけが有効です．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．
-mi	統計情報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．

7.2.5 バッチリクエストの返却

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．リクエスト番号の前に" 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" 211" であった場合，標準出力ファイルは" a.sh.o211" 標準エラー出力ファイルは" a.sh.e211" という名前になります．バッチリクエストファイル名が 8 文字以上の場合は，7 文字で打ち切られて拡張子が付加されます．

! 標準出力ファイルの先頭に

```
Warning: no access to tty (Bad file number).
Thus no job control in this shell.
```

というメッセージが付加される場合があります．これはエラーメッセージではありませんので無視して結構です．

7.3 バッチリクエストの状態表示

7.3.1 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-vpp% qstat          ←バッチリクエストの状態表示
s@kyu-vpp;  type=BATCH;  [ENABLED, INACTIVE];  pri=31
  0 exit;   1 run;   0 queued;   0 wait;   0 hold;   0 arrive;
p1@kyu-vpp;  type=BATCH;  [ENABLED, RUNNING];  pri=31
  0 exit;   2 run;   2 queued;   0 wait;   0 hold;   0 arrive;
        REQUEST NAME       REQUEST ID          USER PRI    STATE  JOB-ID  PHASE
<1 request RUNNING>
    2:           a.sh    213.kyu-vpp         a79999a  31  RUNNING     189  RUN
s8@kyu-vpp;  type=BATCH;  [ENABLED, INACTIVE];  pri=31
  2 exit;   1 run;   2 queued;   0 wait;   0 hold;   0 arrive;
    2:           c.sh    244.kyu-vpp         a79999a  31  RUNNING     189  RUN
                         :
```

7.3.2 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-vpp% qps                ←リクエストの状態表示
que  user     request_ID       cpu       vu    vu/cpu cpu-limit  elapse  v-mem(MB)
p1 a79999a   211.kyu-vpp   0:00:02   0:00:00    0%  20:00:00   0:00:04  384/2048
p8 a79999a   218.kyu-vpp   1:32:11   1:23:23   90%  20:00:00   1:43:04 1984/2048
```

7.4 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．

7.4.1 実行待ちのバッチリクエストのキャンセル

qdel に続けてバッチリクエスト識別子を指定します．

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

7.4.2 実行中のバッチリクエストのキャンセル

-k オプションを指定します．

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

7.5 kyu-cc からのバッチリクエスト

kyu-cc からスーパーコンピュータ VPP5000/64 にジョブを投入することができます．ただし，kyu-cc で作成したソースプログラム，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 ”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-cc と kyu-vpp 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．

7.5.1 バッチリクエストの投入

kyu-cc から kyu-vpp のバッチリクエストを投入する場合には，-q に続けて，キュー名を指定します．p1 キューに投入する場合にも省略することはできません．

```
kyu-cc% qsub -q p1 a.sh     ← p1 キューに投入
Request 338.kyu-cc submitted to queue: p1.
```

7.5.2 バッチリクエストの状態表示

kyu-vpp バッチキューの状態を表示するためには，@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-vpp に投入したリクエストを調べるためには qps コマンドに続けて@kyu-vpp を追加します．

```
kyu-cc% qps @kyu-vpp                ← kyu-vpp のリクエストの状態表示
```

7.5.3 実行待ちのバッチリクエストのキャンセル

-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.
```

7.5.4 実行中のバッチリクエストのキャンセル

-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.
```

8 VPP700/56 からの移行

VPP5000/56 は平成 12 年度末まで運用していたスーパーコンピュータ VPP700/56 の上位互換機であるため，VPP700/56 の資産は基本的にそのまま利用できます．以下に移行に関しての注意点を列挙します．

8.1 実行可能ファイルの指定方法の変更

VPP700/56 では，実行可能ファイル名を，例えば以下のように" a.out "と指定するだけでプログラムを実行することができました．

```
kyu-vpp% a.out              ←実行（VPP700/56 の場合）
```

しかし，セキュリティ強化のため，新 kyu-vpp の推奨する .cshrc ではカレントディレクトリ (現在作業中のディレクトリ) へのパスを設定していません．実行する場合には必ず最初に" ./"を指定してください．

```
kyu-vpp% ./a.out          ←実行（VPP5000/64 の場合）
```

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．また，バッチリクエストファイルも同様に修正が必要になります．

```
#                   ←バッチリクエストの記述例
cd mydir            ←ディレクトリの変更
frt example.f90     ←プログラムの翻訳
./a.out             ←実行（./の指定）
```

8.2 VPP700 のオブジェクトファイルの利用方法

VPP5000/64 で Fortran, C, C++プログラムを翻訳する場合には，省略値として 64 ビットアドレスモードにより処理が行われます．64 ビットアドレスモードによって作成されたオブジェクトファイルと，32 ビットモードである VPP700/56 で作成されたオブジェクトファイルとは互換性がありません．このため，次のようなエラーメッセージが出されることがあります．

```
kyu-vpp% frt main.f90 sub1.o     ← VPP700/56 で作成したオブジェクトファイルとの結合編集
ld: sub1.o: fatal error: wrong machine class
```

オブジェクトファイルのアドレスモードは，file(/usr/bin/file) コマンドによって調べることができます．

```
kyu-vpp% file sub1.o          ←ファイル型の判定
sub1.o:       ELF 32 ビット MSB 再配置可能 VPP Version 1
```

VPP700/56 で作成したオブジェクトファイルと結合するためには，frt, cc, CC コマンドのオプションとして-KA32 を指定し，32 ビットアドレスモードでの翻訳に切替えます．

```
kyu-vpp% frt -KA32 main.f90 sub1.o        ← 32 ビットアドレスモードでの翻訳・結合編集
```

ただし，32 ビットアドレスモードで作成した実行可能ファイルは記憶容量の上限が 2GB に制限されます．2GB 以上の記憶容量を使用する場合には，VPP700/56 で作成したソースファイルを省略値である 64 ビットアドレスモードで再コンパイルする必要があります．

8.3 C コンパイラ起動コマンドの変更

VPP700/56 では C のスカラー翻訳コマンドが cc，ベクトル翻訳コマンドが vcc でした．VPP5000/64 では cc コマンドに統一され，"-Kvp" オプションを指定することによりベクトル翻訳が行われます．

8.4 Fortran の書式なし入出力文

Fortran 95 仕様からは書式なし入出力を行う場合に OPEN 文に FORM='UNFORMATTED' を指定していないとエラーとなります．

```
REAL(KIND=8),DIMENSION(100) :: X                          ! 配列の宣言
   :
OPEN(1,FILE='EXAMPLE.DATA',FORM='UNFORMATTED')            ! 書式なしデータを開く
READ(1) X                                                 ! 書式なし入力
CLOSE(1)                                                  ! ファイルを閉じる
```

8.5 Gaussian 98 の環境設定

VPP700/56 で Gaussian 98 を利用されていた方は，kyu-vpp に接続する際に以下のメッセージが出力されることがあります．

```
/usr/local/gaussian98/g98/bsd/g98.login: Not a directory.
```

VPP5000/64 では，Gaussian 98 のレベルアップにともない環境設定の方法が変更になりました．お手数ですが，

http://www.cc.kyushu-u.ac.jp/scp/system/library/Gaussian/Gaussian98.html

を参考に利用者自身で設定変更をお願いします．

8.6 バッチジョブ終了後の使用額通知メールの廃止

これまで kyu-vpp に投入したバッチジョブが終了すると，kyu-cc 宛にジョブの使用額が送信されていました．VPP5000/64 では，現在のところ，使用額通知機能が未サポートです．

現在の使用額は utlist(/usr/local/bin/utlistf) コマンドによって調べることができます．ただし集計が完了していない課金対象資源もありますので utlist コマンドで表示される額は目安とお考えください．

8.7　ベクトル演算の丸めモード

VPP700/56 のベクトル演算の丸めモードは「0 方向丸めモード」でした．VPP5000/64 のベクトル演算の丸めモードは IEEE754 規格に準拠した「最近値丸めモード」です．このため，丸め誤差の影響を受けやすいプログラムでは実行結果に違いが生じる可能性があります．なお，スカラー演算では VPP700/56, VPP5000/64 ともに「最近値丸めモード」です．

8.8　実行結果に精度差が生じる可能性

上記浮動小数点演算の丸めモードの違いの他に，ハードウェアの違いによって，総和演算の演算順序，複合演算， 組込み関数の性能が VPP700/56 と VPP5000/64 とでは異なります．そのため，実行結果に精度差の生じる可能性があります．

参考文献

[1] Ellis, M. A. and Stroustrup, B.: The Annotated C++ Reference Manual, Addison Wesley (1990).（足立高徳, 小川祐司訳: 注解 C++ リファレンスマニュアル, トッパン）

[2] Kernighan, B. W. and Ritchie, D. W.: The C Programming Language, Prentice Hall (1998).（石田晴久訳: プログラミング言語 C, 共立出版）

[3] 九州大学情報基盤センター スーパーコンピュータ オンラインマニュアル [‡], http://www.cc.kyushu-u.ac.jp/scp/system/manual/UXPV_MANUAL/

[4] 南里豪志: 情報基盤センター・研究者用システムでの UNIX 利用法, UNIX 初級講習会資料 (2001).

- PDF ファイル (449,738 bytes)“ unixguide.pdf ”
  http://spring.cc.kyushu-u.ac.jp/scp/system/library/UNIX/unixguide.pdf
- PostScript ファイル (600dpi; 1,030,609 bytes)“ unixguide.ps ”
  http://www.cc.kyushu-u.ac.jp/scp/system/library/UNIX/guide1.ps
- PostScript ファイル・gzip 圧縮 (600dpi; 196,824 bytes)“ unixguide.ps.gz ”
  http://www.cc.kyushu-u.ac.jp/scp/system/library/UNIX/guide1.ps.gz

[5] 南里豪志: 新スーパーコンピュータシステムの紹介, 九州大学情報基盤センター広報, Vol. 1, No. 1, pp.20–30 (2001).

[‡]本センターの利用資格を有している方しか参照できません．登録番号とパスワードを尋ねてきますので，登録番号のみを入力してください．パスワードは空白で構いません．