スーパーコンピュータ SX-3R 利用法

システム管理掛　青井信一
システム管理掛　中島重雄

1　まえがき

大阪大学大型計算機センターに NEC のスーパーコンピュータ SX-3 モデル 14R(以下 SX という) が導入され 93 年 2 月 1 日から利用可能となりました。この SX を利用する方法として，汎用計算機 ACOS2020(以下 ACOS という) から利用する方法と UNIX ワークステーションから利用する 2 つの方法が用意されています。ここでは，これから ACOS を経由して SX を利用しようとされる初心者の方，ならびに今までワークステーションを利用して UNIX を使われていた方を対象に，SX を利用する方法を簡単に説明します。ACOS を経由して使われる方は第 3章「SX の利用形態」および第 4章「簡易形 TSS コマンドの使用法」，第 5章「バッチジョブの使用法」を，ワークステーションを経由して利用される方は，第 3章の「SX の利用形態」と第 5章「バッチジョブの使用法」をご覧下さい。

2　SX の機器構成

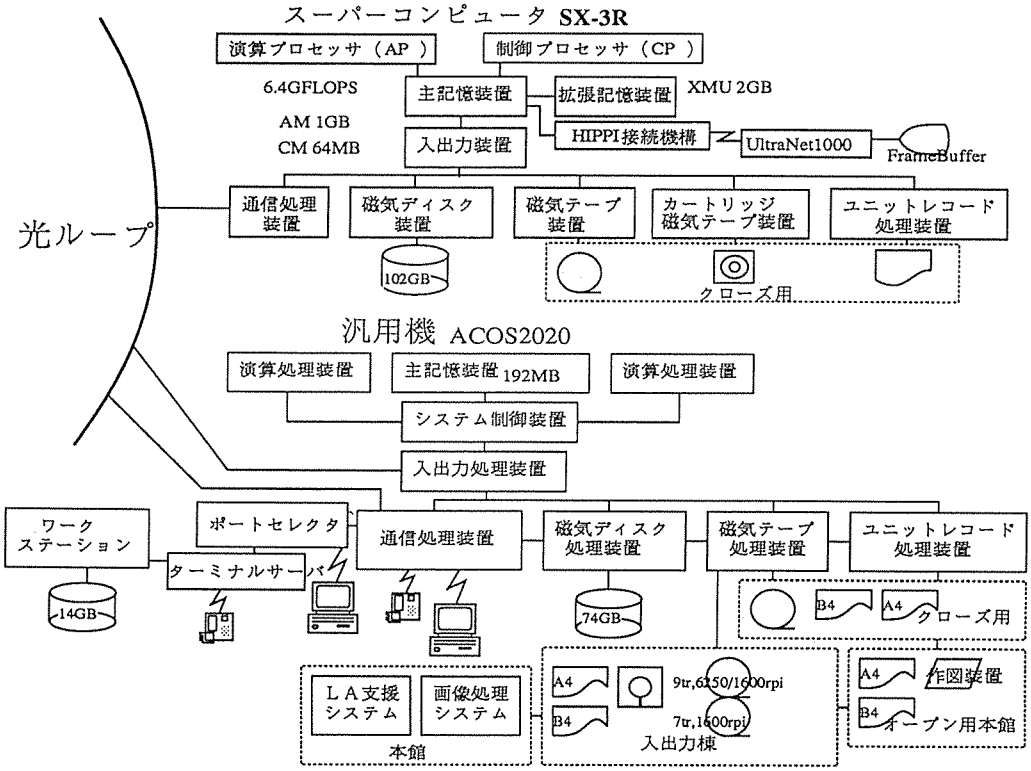

図 2.1: 機器概要

導入されたスーパーコンピュータは最大ベクトル演算性能6.4GFLOPS，主記憶装置1GB，拡張記憶装置2GBを備え，主な周辺装置として磁気ディスク装置102GB，ならびに動画像をリアルタイムに表示させる動画像表示装置があります。

SXの点線内は，運用管理のための装置で，利用者の方は使用できません。このSXには演算処理装置にAP（Arithmetic Processor）とCP（Control Processor）があります。CPは主に入出力処理制御を担当し，APがFortranコンパイラや利用者プログラムの実行を担当します。

3 SXの利用形態

SXを使用する利用形態として，ACOSあるいはワークステーションからSXへ直接接続して会話型で利用できる会話型直接利用形，センターのワークステーションからSXへ直接バッチジョブを投入できるNQS[1]形，SXへは直接接続しないでACOSからSXを利用する簡易形と基本形，の4つがあります。UNIX利用者の方はぜひ一度このNQS形でスーパーコンピュータの能力を確かめて下さい[2]。

なお，このSXにはオペレーティングシステム(以後OSと記述)としてAT&T UNIX System Vに準拠したSUPER-UX(4.3BSD機能組み込み)が採用されています。汎用機ACOSにはNEC独自のACOS-6が搭載されています。

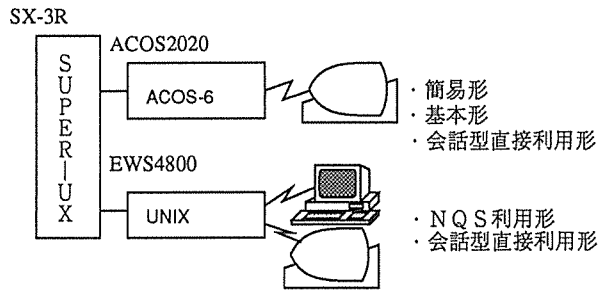

図 3.1: スーパーコンピュータの利用形態

3.1 会話型直接利用形

UNIX本来の利用方法で，ACOSあるいはワークステーションから直接SXへ接続し，会話型で利用する方法です。ACOSあるいはワークステーションを経由しますが実行するのはあくまでもSUPER-UXのコマンドです。SUPER-UXが提供している全ての機能が利用できます。ただし，

[1]NQS(Network Queuing System)：ネットワークで結ばれたUNIXシステム間でバッチ処理を行います。

[2]簡易形ではCPU時間，経過時間しか分かりません。メモリサイズがどれくらいなのか，ベクトル演算率がどれ位なのかは分かりません。基本形/NQS形ではtimexコマンドを使用しオプションとして`-p -t -M`を指定すればCPU時間，最大使用メモリ量が出力されます。コマンド形式 `timex -p -t -M` *command*

ACOS 経由の場合は利用できない機能があります。

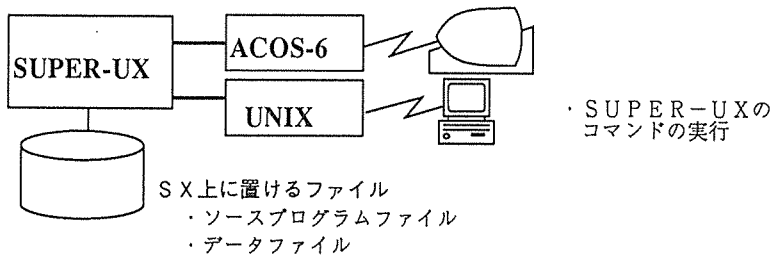

図 3.2: 会話型

3.2 NQS 形

センターに設置されているワークステーションから SX にバッチジョブを投入する方法[3]です。UNIX の NQS を利用してバッチジョブを投入することをバッチリクエストともいいます。プログラムおよびデータはワークステーションあるいは SX どちらに置かれていても構いませんが，ワークステーション上にある場合は，SX にコピーあるいは転送する必要があります。

ジョブの投入問い合わせ等については NQS 用のコマンドを使用します。

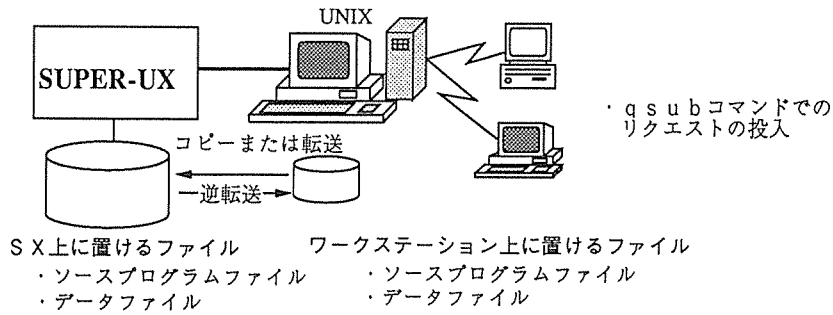

図 3.3: NQS 形

3.3 基本形

ACOS から SX をバッチジョブで利用する方法です。SX 上にソースプログラムおよびデータファイルを持ち，実行する一連の手順をシェル[4]コマンドで作成し ACOS 上にファイルとして持ち，そのファイルをバッチジョブとして SX に投入します。シェルプログラムを作成するには UNIX

[3]センターには SPARC，IRIS，EWS4800 が設置されていますが，NQS が利用できるのは当面 EWS のワークステーションだけです。

[4]シェルは，UNIX オペレーティングと通信できる唯一のプログラムであり，利用者とオペレーティングシステムの核との間の連絡を行い，利用者が入力したコマンドを解析し OS に実行を依頼します。また，条件実行および制御フローを有するプログラミング言語としての機能も持っています。代表的なものに C シェル，B シェルがあります。

の基礎的な知識が必要となります。もちろん，ACOS とのファイル転送コマンドが用意されていますので，ACOS 上のソースプログラムおよびデータを SX に転送して利用することもできます。簡易形と同じくジョブは SX に転送され，ジョブ終了時に結果は自動的に逆転送されますが，使用しているファイルは SX 上に保存されたままです。SX 上のファイルを ACOS に逆転送することも可能です。

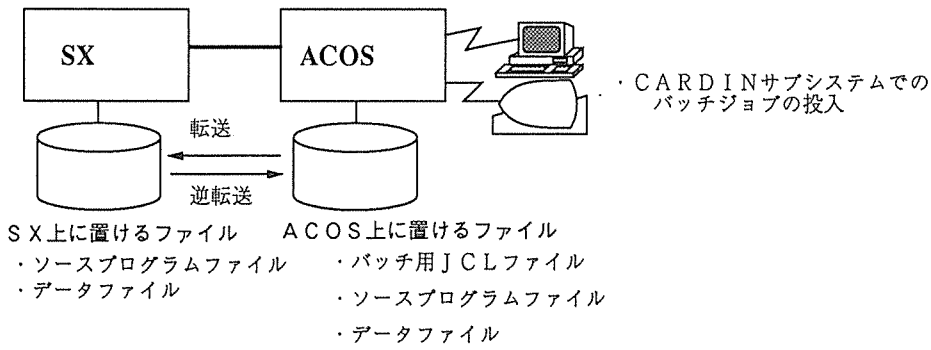

図 3.4: 基本形

UNIX 利用者の方には比較的簡単に SX を利用できるものと思われますが，ファイル作成/変更などが会話的にできないため，UNIX を使いなれた方にとっては利用しにくい形態だと思われます。

なお，基本形では ACOS からバッチジョブを投入する方法のみで，会話型では利用できません。

3.4 簡易形

従来からの ACOS 利用者のために，ACOS のジョブ制御言語[5]（以後 JCL と記述）および TSS コマンド[6]の形式に合わせて作成された，ACOS から SX を利用する方法で，SX へ接続する操作は必要はありません。

簡易形には利用者が JCL で作成したジョブをカードインサブシステムで投入する方法，SXJOB コマンドでバッチジョブを投入する方法，および SXRUN コマンドで利用者のプログラムの翻訳/実行のみを SX 上で会話的に行う 3 つの方法があります。

ソースプログラムは必ず ACOS 上のファイルで，データは ACOS,SX どちらに作成されていても構いません。ACOS 上の指定されたファイルが SX の作業用ファイルに自動的に転送され，処理終了時に必要に応じて逆転送されます。SX 上にあるデータファイルを使用する場合は UNIX のファイル構造[7]を知っておく必要があります。

ただし，簡易形では翻訳/結合/実行の一連の処理でのみ利用可能で，実行だけを行うということはできません。実行だけを行いたい場合は，基本形あるいは NQS 形を利用していただくことに

[5]計算機に利用者の仕事を実行させるための命令
[6]JCL と同じく計算機に利用者の仕事を実行させるための命令。JCL と違い会話的に利用できる。
[7]「スーパコンピュータ利用の手引き」(基本形編) をご覧下さい。

なります。

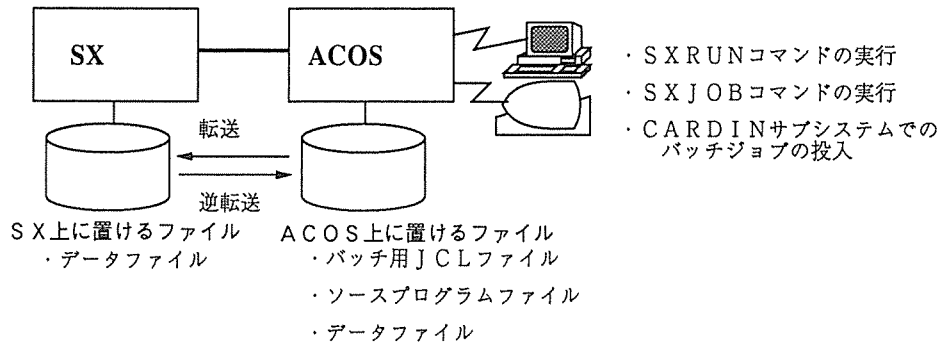

図 3.5: 簡易形

3.5 利用できる主なソフトウェア

利用できる主なソフトウェアを次に掲げます。

- プログラム言語
  - ◇ FORTRAN77/SX：JIS FORTRAN(X3001-1982) に準拠した言語。
  - ◇ C/SX：ANSI C(X3J11/90-013) に準拠した言語。
  - ◇ C++：C 言語をベースにした汎用プログラミング言語。
- 性能向上支援ツール
  - ◇ ANALYZER/SX：FORTRAN77 ソースプログラムのベクトル化に関するチューニングのための静的/動的特性を解析する性能向上支援ツール。
- デバッグ支援ツール
  - ◇ dbx：C,FORTRAN 言語で記述されたプログラムをソースレベルのシンボルの名前や行番号でデバッグできるシンボリックデバッガ。プログラムの記述言語に依存せず，機械語レベルでのデバッグも可能。会話型でしか利用できません。
  - ◇ sdb：C 言語および FORTRAN77/SX で作成されたプログラムを言語レベルで扱うシンボリック・デバッガ。デバッグされるプログラムとソースレベルで会話することができる。会話型でしか利用できません。
  - ◇ make コマンド：プログラム群の保守，更新および再生成を行う。
- ライブラリ
  - ◇ 科学技術計算ライブラリ
    高速性と広い適応性を特徴とするベクトル化された科学技術計算ライブラリ IMSL, ASL/SX。

◇ 数値計算ライブラリ
数値計算や統計計算に必要な多くの手法が集められたベクトル化された数値計算ライブラリ MATHLIB/SX。

◇ グラフィックライブラリ
国際標準の2次元グラフィックサブルーチンライブラリ GKS，および国際標準の3次元グラフィックサブルーチンライブラリ PHIGS。

ただし，簡易形，基本形，NQS 形，会話型で使用できるソフトウェアが異なります。表3.1をご覧下さい。簡易形では，FORTRAN 言語しか利用できませんが，基本形，NQS 形では原則的に SUPER-UX が提供する全てのソフトウェアが利用できます。ただし，端末との入出力を行うソフトウェアは利用できません。会話型では全てのソフトウェアが利用できますが，ACOS 経由の場合は利用できないソフトウェアがあります。

表 3.1: 利用できる主なソフトウェア

ソフトウェア		簡易形	基本形/NQS 形	会話型
プログラム言語	FORTRAN77/SX	○	○	○
	C/SX	×	○	○
	C++	×	○	○
性能向上支援ツール	ANALYZER/SX	○	○	○
デバッグ支援ツール	dbx	×	×	○
	sdb	×	×	○
	make	×	○	○
ライブラリ	IMSL	○	○	○
	ASL/SX	○	○	○
	MATHLIB/SX	○	○	○
	GKS	×	×	○
	PHIGS	×	×	○

○：利用可，　×：利用不可

4　簡易形 TSS コマンドの使用法

SXRUN と SXJOB の2つのコマンドがあります。SXRUN は FORTRAN プログラムの翻訳/実行だけを SX 上で会話的に行うものであり，SXJOB はバッチジョブを SX へ投入するコマンドです。どちらも ACOS 上で実行するコマンドであり，SX へ接続する操作は必要ありません。

4.1　SXRUN コマンド

ACOS 上にあるソースプログラム/データを SX 上の一時ファイルへ自動的に転送し，会話的に翻訳/実行を行います。データの入出力は端末から直接入出力することもファイルから入出力す

ることも可能です。SX上に作成されているデータファイルはFORTRANプログラム中のOPEN文で割り当てることができます。

SYSTEM選択レベルあるいはビルドモードでSXRUNコマンドを入力します。

SYSTEM?SXRUN ソースプログラムファイル：オプション

なお，SXRUNではCPU時間10分，メモリサイズは40MBまでしか利用できません。次に代表的な使い方の例を紹介します。

最初は一般的と思われる，コンパイルオプションをすべて既定値でデータをファイルから入力している例です。

例

```
SYSTEM?SXRUN␣W60000/FRT3:F=W60000/DATA/D01,R(05)
  INPUT   DATA=1     RESULT=     19
  INPUT   DATA=5     RESULT=     10
SYSTEM?
```

利用者番号W60000直下のソースプログラムファイル

FRT3 ファイル

を翻訳/実行しています。もちろん，エラーがあれば実行されません。
通常，オプション[8]はすべて規定値で問題ありませんが，ソースプログラムの記述形式の既定値がSXRUNでは**FREE2**ですので，この例のソースプログラムは

自由形式2

で書かれている必要があります。
実行時にはプログラムの中でデータをREAD(5,･･･で読み込んでいる場合が多いと思われますが，入出力のデータはFオプションで割り当てます。

F=W60000/DATA/D01,R(05)

ここでは，利用者番号W60000の下のカタログ[9]DATAの下のファイル[10]D01が機番05の入力データとなっています。WRITE(6,･･･が使用されていれば，結果は端末に出力されます。

多くのオプションを使用する場合，いちいちオプションを指定するのはめんどうですので，コマンドとオプションをファイルに埋め込んで実行できます。計算結果も端末に出力するのではなくファイルに出力することも可能です。

ファイルの先頭に*#に続きSXRUNコマンドとオプションを記述します。

[8]形式と詳細な説明はセンター発行資料「スーパーコンピュータ利用の手引き(簡易形編)」をご覧下さい。
[9]ファイルを管理/識別するための記述子。階層構造をとることができる。
[10]利用者のデータが実際に格納されている。

例

```
*SXRUN␣/FRT4
*
```

ファイル FRT4 の内容

```
*#SXRUN␣/FRT4:FIXED␣F=/DATA01,R(05)␣F=/DATA11,W(06)
C␣␣␣␣␣TEST␣PROG.
␣␣␣␣␣␣INTEGER␣DATA05,RESULT
␣␣␣␣␣␣READ(5,10)A,B
␣␣␣10␣FORMAT(...
␣␣␣␣␣␣WRITE(6,100)RESULT
␣␣100␣FORMAT(...
```

ACOS と接続したときの利用者番号の下のファイルを利用する場合は，上の例のように利用者番号を省略できます。

オプションが一行に書ききれない場合は最後に; を書き次行に継続することができます。

```
*#SXRUN␣/FRT4:F=/DATA01,R(05)␣F=/DATA11,W(06);
*#FIXED␣INLINE␣MASK=(FLUNF)
 C␣␣␣␣␣TEST␣PROGRAM
␣␣␣␣␣␣INTEGER␣INDATA,OUTDATA
␣␣␣␣␣␣READ(5,10)ABC
␣␣␣10␣FORMAT(...
␣␣␣␣␣␣WRITE(6,100)RE
```

センター提供のライブラリには IMSL 社の数値科学計算用 IMSL ライブラリ，および NEC の科学技術計算ライブラリ ASL と数値計算ライブラリ MATHLIB があります。これらのライブラリを使用するには L オプションでライブラリファイルを割り当てます。

```
SYSTEM?SXRUN␣/FRT5:L=/LIB/ASL
```

ここでは ASL ライブラリを割り当てています。ライブラリの割当は必ず/から始め大文字で指定します。指定するライブラリは表 4.1をご覧下さい。

表 4.1: ライブラリファイル名

ライブラリファイル名	ライブラリ名称
/LIB/IMSL	IMSL ライブラリ
/LIB/ASL	科学技術計算ライブラリ ASL/SX
/LIB/MATH	数値計算用ライブラリ MATHLIB/SX

4.2 SXJOB コマンド

ACOS 上にあるソースプログラム，データファイルから一つのバッチジョブを作成し，SX に投入します。このとき SX に投入されたバッチジョブは UNIX で使用される NQS シェルに変換されます。簡易形利用者の方はこの NQS シェルを特に理解する必要も，後で説明します簡易形の JCL を作成/理解する必要もありませんので，SX へ簡単にバッチジョブが投入できます。

結果の検索/出力はACOS側でJOUTコマンドを用います。ジョブの実行状況はJSTS, JMONIコマンドで確認できます。

SYSTEM選択レベルあるいはビルドモードでSXJOBコマンドを入力します。

SYSTEM?SXJOB ソースプログラムファイル:オプション

次の例は，FORTRANのコンパイルオプションにすべて既定値を採用し，実行時のデータ入力READ(5,···) をファイルに割り当てている例です。

例

```
SYSTEM?SXJOB␣/PROG/FRT1:JOBCLS=U␣F=/DATA/D05,R(05)
SNUMB#0001T
SYSTEM?
```

ACOSのTSSと接続した利用者番号の下のカタログPROGの下のファイルFRT1に格納されているプログラムを投入しています。このファイルはJCLやデータを含まないソースプログラムだけが格納されている必要があります。複数のファイルに分かれている場合は，空白で区切ってファイル名をならべます。

```
SXJOB␣/PROG/MAIN␣/PROG/SUB1
```

このSXJOBコマンドにはただ一つ必須のオプションがあります。

JOBCLS=U

オプションの**JOBCLS**が必須であり，ここでは**U**を指定しています。このオプションはジョブが使用する資源の大きさによって表4.2のようにクラス分けされています。

表 4.2: ジョブクラス

ジョブクラス	CPU時間	プログラムサイズ	作業用ファイル容量
U	10分	40MB	100MB
V	30分	200MB	
W	2時間	500MB	
会話型	10分	40MB	

MB:Mega Bytes, 1バイト1文字

次に使用している

F=/DATA/D05,R(05)

オプションはプログラム中の外部装置識別子05（READ(5,nn)･･･）をファイルに割り当てています。ACOSのTSSと接続した利用者番号の下のカタログDATAの下のファイルD05をREAD(5,nn)･･･に割り当てています。

複数のオプションを指定するときは空白で区切ります。

ジョブが受け付けられるとシステム側からジョブに割り振られた番号(SNUMB番号)が出力されます。

バッチジョブのプログラムの記述形式は，既定値が**FIXED**になっています。プログラムが自由形式2で記述されている場合は

FREE2

オプションが必要になります。

WRITE(6,nn)･･･の結果はプリントイメージとしてACOS側に返ってきますが，ファイルに出力することもできます。

例

```
*SXJOB␣/FRT2:F=/DATA/D01,R(01)␣F=/D20,W(06)␣FREE2␣JOBCLS=W
 SNUMB#0002T
*
```

F=/DATA/D01,R(01)

このオプションはプログラム中の外部装置識別子01（READ(01,nn)･･･）をカタログDATAの下のファイルD01から入力し，

F=/D20,W(06)

は外部装置識別子06（WRITE(06,nn)･･･）のデータを利用者番号直下のファイルD20に出力しています。

プログラムが自由形式2で記述されていますので，

FREE2

オプションを指定しています。

また，ジョブクラスにはWを指定していますので，

JOBCLS=W

演算時間は2時間，プログラムの大きさは500MBまで利用できます。2時間以上を必要とする場合は，ご面倒ですがプログラムを分割していただかなくてはなりません。SXは最大ベクトル演算性能6.4Gflops，スカラー性能でSX-2Nの約2.2倍ですので2時間でかなり大きな計算ができると思われますが。

オプションが1行に書ききれない場合は，オプションの最後にセミコロン（；）を入力しますと継続の問い合わせが行われます。

例

```
SYSTEM?SXJOB␣/FRT3:F=/DATA/D01,R(01)␣F=/D20,W(20)␣JOBCLS=V;
SXJOB001_R_CONTINUE?LIST=ALL␣L=IA␣SOURCE␣SUMMARY
SNUMB#0004T
SYSTEM?
```

この例では，1行目の最後が；で終わっていますので，オプションの追加の入力促進記号が2行目に出力され

SXJOB001 R CONTINUE?

入力待ちとなっています。
2行目では，まず最初にJCLイメージ，変換されたNQSイメージなどすべてのレポートを出力するオプション

LIST=ALL

を指定しています。このイメージが不要であればオプションを省略して下さい。
次に，センターで提供しているライブラリIMSLを使用しています。

L=IA

もちろん，ライブラリを使用しなければこのオプションも指定する必要はありません。利用できるライブラリには表4.3のものがあります。Lオプションにはライブラリコードを指定します。

表 4.3: ライブラリ

ライブラリコード	ライブラリ名称
IA	IMSL ライブラリ
AA	科学技術計算ライブラリ ASL/SX
MA	数値計算用ライブラリ MATHLIB/SX

次に，FORTRANコンパイラのオプションでソースイメージとサマリリストを出力する

SOURCE と SUMMARY

を指定しています。

実行データがない場合のプログラムを実行するときは，RUNオプションが必要となります。

例

```
SYSTEM?SXJOB␣/PROG/FRT2:RUN␣JOBCLS=W
SNUMB#0005T
SYSTEM?
```

RUN オプションが指定されなかった場合，ジョブは翻訳と結合処理のみが行われ実行は行われません。ただし，F オプションで実行時のデータが割り当てられている場合，RUN オプションが省略されていても実行されます。

学外の利用者の方で計算結果を郵送したい場合は，

JOBID=MAIL

を指定します。センターのプリンタに自動的に出力され，郵送されます。

例

```
SYSTEM?SXJOB␣/PROG/FRT2:JOBCLS=W␣JOBID=MAIL
SNUMB#0006T
SYSTEM?
```

計算結果をセンターの A4 サイズのプリンターに出力したい場合は，

OUTUNIT=JPA4

を指定しておきます。この他に JPB4(連続紙),CPA4(カット紙) が指定できます。

例

```
SYSTEM?SXJOB␣/PROG/FRT2:JOBCLS=W␣OUTUNIT=JPA4
SNUMB#0007T
SYSTEM?
```

ジョブが投入されると，システム側からジョブを識別するための受付番号 (SNUMB) が出力されます。問い合わせ/検索などにはこの受付番号を使用しますので忘れないで下さい。

なお，ファイルを割り当てる場合，カタログポジショニング機能[11]は利用できません。コマンドは正常に終了しても作成されたバッチジョブでのファイル割当はエラーとなります。また，指定したファイルが TSS で使用中の場合，' ILLEGAL FILE DESCRIPTION ' エラーが発生し SX へ投入できません。ご注意下さい。

5 バッチジョブの使用法

簡易形/基本形では利用者の方が JCL を作成する必要[12]があり，NQS 形では JCL を作成する必要はありませんが，2,3 の NQS コマンドならびにオプションを使用する必要があります。簡易形/基本形は ACOS 上から，NQS 形はワークステーション上から投入します。JCL の詳細な説明は「スーパーコンピュータ利用の手引き」(簡易形編)，(基本形編) を，NQS 形については「スーパーコンピュータ利用の手引き」(NQS 編) をご覧下さい。

[11] カタログを省略形で利用する機能。
[12] SXJOB コマンドでは不要です。

5.1 バッチジョブの構成

簡易形のバッチジョブではFORTRANプログラムを翻訳/結合し，実行する，というジョブ構成でしか利用できません。もちろん翻訳処理のみでも可能です。基本形では，様々な利用が可能です。簡易形/基本形のジョブは利用者宣言文であるJOB文で始まり，ジョブ終了宣言文であるENDJOB文で終わります。JOB，NQS，ENDJOB文はジョブを構成する基本的なJCL文です。簡易形ならびに基本形を使用する場合は必須の制御文で，1カラム目に$，8カラム目から制御文名を，16カラムからそのオプションを記述します。

ジョブの先頭には利用資格のチェックおよび利用者の識別を行う$JOB文を書きます。16カラムから；に続き支払コードおよびジョブクラスを指定します。

```
$␣␣␣␣␣␣␣JOB␣␣␣␣␣␣;A,U
```

支払コードにはセンターに申請してある支払費目のうち，ジョブ実行に要した負担金を支払う費目を指定します。指定できるジョブクラスについては表4.2を参照下さい。この二つのオプションは必須です。
続いて学外の方が計算結果を郵送する場合は**MAIL**を指定します。

```
$       JOB             ;A,U,MAIL
```

計算結果をセンターのプリンタに出力する場合，第6オプションに出力装置を指定します。

```
$       JOB             ;A,U,,,,JPA4
```

この例ではA4サイズの連続紙のプリンタを指定しています。B4サイズの連続紙の場合は**JPB4**，A4サイズのカット紙の場合は**CPA4**となります。オプションの順序は決まっており，省略する場合は必ずカンマ(，)を記述します。

JOB文の次には簡易形/基本形の宣言を行う$NQS文を置きます。

```
$␣␣␣␣␣␣␣NQS␣␣␣␣␣␣TYPE=A6,LIST=ALL
```

16カラムからのオプションが簡易形では**TYPE=A6**，基本形では**TYPE=UX**となります。既定値は基本形ですので，簡易形の方がこのオプションを省略すると，実行はされますが‘そのようなコマンドはない’というエラーが出力されます。御注意下さい。
その次のオプションはSX用に変換されたシェルのイメージを出力するようにしています。変換されたシェルのイメージが不要な場合は省略できます。

ジョブの最後には$ENDJOB文を置きます。

```
$␣␣␣␣␣␣␣ENDJOB
```

簡易形では基本的には次のように$NQSと$ENDJOB文の間にプログラム，データおよび簡易形用JCLを挿入します。

例

```
$␣␣␣␣␣␣␣JOB␣␣␣␣␣␣;A,U,,,,JPA4
$␣␣␣␣␣␣␣NQS␣␣␣␣␣␣TYPE=A6
          JCL，ソースプログラム
          およびデータ
$␣␣␣␣␣␣␣ENDJOB
```

基本形では$NQSと$ENDJOB文の間にUNIXのコマンドならびにシェルコマンドを記述します。利用者はUNIXの使い方を知っている必要があります。

例

```
$␣␣␣␣␣␣␣JOB␣␣␣␣␣␣;A,U,,,,JPA4
$␣␣␣␣␣␣␣NQS␣␣␣␣␣␣TYPE=UX
          シェルスクリプト
             .
$␣␣␣␣␣␣␣ENDJOB
```

NQS形では簡易形/基本形と少し異なりJCLを記述する必要はありませんが，シェルスクリプトを作成しNQS用オプションを先頭に埋め込むか，ジョブ投入時にオプションを指定する必要があります。

例

```
#      オプション列
#
          シェルスクリプト
```

今まで例で使用した$JOB文の例は大学間ネットワークでは利用できません。大学間ネットワークを利用して阪大センターを利用する場合は次のように記述して下さい。

$␣␣␣␣␣␣␣JOB␣␣␣␣␣␣利用者番号;支払コード$パスワード,ジョブクラス,RMT,,,JPA4

5.2 簡易形/基本形用ジョブ

簡易形と基本形のジョブの作成と投入/結果の受取を説明します。

5.2.1 簡易形用ジョブの作成

簡易形用JCL文には，FRT77，GO，PRMFL，FILEなどがあります。もちろん，これらは基本形では使用できません。

$FRT77文はNQS文の次におき，FORTRAN77/SXコンパイラを呼び出し，翻訳/結合処理を行います。

次の例はオプションを使用せず，既定値を採用して翻訳/結合を行っています。

```
$␣␣␣␣␣␣␣FRT77
```

この場合，バッチジョブのプログラムの記述形式の既定値がFIXEDですのでプログラムは，

固定形式

で記述されている必要があります。自由形式で書かれている場合は FREE，自由形式2で書かれている場合は FREE2 オプションを指定する必要があります。

次の例はソースリストとサマリリストを出力するオプションを使用しています。オプションはカンマ (,) で区切ります。

```
$␣␣␣␣␣␣␣FRT77␣␣␣SOURCE,SUMMARY
```

この FRT77 文の後にソースプログラムを置きます。$SELECTA 文あるいは$PRMFL 文が使用できます。もちろん，直接プログラムを書くこともできます。

```
$␣␣␣␣␣␣␣SELECTA␣W60000/KATA/FAIRU
```

この例では利用者番号 W60000 の下のカタログ KATA の下のファイル FAIRU の内容がこの JCL が挿入された位置に展開されます。

$SELECTA 文の代わりに$PRMFL 文を使用することもできます。

```
$␣␣␣␣␣␣␣PRMFL␣␣␣S*,R,S,W60000/PROG
```

1番目のオプションは，ソースプログラムを割り当てる場合は“S*”と決まっています。2番目は読み込みか書き込みかを指定するもので，ソースプログラムなので読み込みの“R”，3番目はファイルの形式を指定するもので順編成の“S”，4番目には割り当てるファイルを指定します。この例ではソースプログラムとして利用者番号 W60000 の下のファイル PROG を割り当てています。$SELECTA 文と$PRMFL 文との大きな違いは，$SELECTA 文は SX にジョブが投入されるときに内容が展開されるのに対し，$PRMFL 文はジョブが SX で実行に入ってから内容が転送されます。したがって SX で実行に入ったときファイルが ACOS 側で使用されていれば転送エラーとなります。もう一つの違いは，LIST=ALL オプションを使用した時，$SELECTA 文であれば変換された NQS シェルイメージの中にソースイメージが埋め込まれ表示されます。

利用者プログラムを翻訳/結合してエラーがなければ$GO 文で実行します。

```
$␣␣␣␣␣␣␣GO
```

センターが提供しているライブラリを使用したい場合は，この$GO 文のオプションにライブラリオプション名を指定します。ライブラリオプション名については表5.1をご覧下さい。

```
$␣␣␣␣␣␣␣GO␣␣␣␣␣␣␣IMSL,ASL
```

この例では，IMSL ライブラリと科学技術計算ライブラリ ASL を使用することを宣言しています。

次の例は実行時にデータを必要としないジョブのイメージです。

表 5.1: ライブラリ名

ライブラリオプション名	ライブラリ名称
IMSL	IMSL ライブラリ
ASL	科学技術計算ライブラリ ASL/SX
MLIB	数値計算用ライブラリ MATHLIB/SX

例

```
$␣␣␣␣␣␣␣JOB␣␣␣␣␣␣;A,W,,,,JPA4
$␣␣␣␣␣␣␣NQS␣␣␣␣␣␣TYPE=A6
$␣␣␣␣␣␣␣FRT77
$␣␣␣␣␣␣␣SELECTA␣W60000/PRO/MAIN
$␣␣␣␣␣␣␣SELECTA␣W60000/PRO/SUB
$␣␣␣␣␣␣␣GO
$␣␣␣␣␣␣␣ENDJOB
```

実行時にデータを使用する場合は前述の$SELECTA 文あるいは$PRMFL 文で割り当てます。READ(05,nn)･･･のデータは$SELECTA 文でも構いませんが 05 以外の機番の場合は$PRMFL 文を使用しなければなりません。

例

```
$     JOB     ;A,W,,,,JPA4
$     NQS     TYPE=A6
$     FRT77
$     SELECTA W60000/PRO/MAIN
$     SELECTA W60000/PRO/SUB
$     GO
$     SELECTA W60000/DT/DATA1
$     PRMFL   10,R,S,W60000/DT/DATA10
$     ENDJOB
```

実行時のデータを$SELECTA 文と$PRMFL 文で割り当てています。利用者番号 W60000 の下のカタログ DT の下のファイル DATA1 が READ(05,nn)･･･の入力データとなり，ファイル DATA10 が READ(10,nn)･･･の入力データとなります。

プログラム中で結果を WRITE(06,nn)･･･以外の機番で使用している場合，その機番をファイルに割り当てることができます。

```
$       PRMFL    20,W,S,W60000/DATA
```

1番目のオプションはFORTRANプログラムの外部装置識別子に対応します。この例ではプログラム中の機番20に対応します。
2番目のオプションはファイルの属性を指定します。書き込みデータの場合W，読み込みデータの場合Rを指定します。書き込みデータの時に誤ってRを指定した場合，実行はされますがデータは書き込まれません。
3番目のオプションはファイルの形式を指定しますが，簡易形では順編成形式しか利用できませんので，必ず，Sとなります。
4番目のオプションは外部装置識別子に対応させるファイルを利用者番号から指定します。
この例ではW60000の利用者番号の下のファイルDATAに20番の機番が割り当てられることになります。

ただし，転送あるいは逆転送できる1レコードの長さに制限[13]があります。順編成の場合は1レコードの長さが32,768バイト迄しか転送/逆転送できません。制限を超えると‘レコード長エラー’が出力され実行されません。

複数の外部装置識別子を使用している場合は，それぞれに$PRMFL文が必要です。

例

```
$       JOB     ;A,W,,,,JPA4
$       NQS     TYPE=A6
$       FRT77
C   MAIN PROGRAM
C
         .
       READ(5,10)A,B
   10  FORMAT(
       WRITE(6,100)HIST,DATA
       WRITE(10,200)
       WRITE(20,1000)RESULT
 1000  FORMAT(
         .
       END
$       GO
$       PRMFL   05,R,S,W60000/INDATA
$       PRMFL   10,W,S,W60000/OUT/DATA1
$       PRMFL   20,W,S,W60000/OUT/DATA2
$       ENDJOB
```

これまでのファイルの割当はパーマネントファイルに割り当てていましたが，一時ファイルを使用することもできます。

[13] 93年3月1日現在の制限です。機能強化で対処予定。

```
$␣␣␣␣␣␣FILE␣␣␣␣10,,L
```

外部装置識別子 10 番を順編成形式で一時ファイルに割り当てています。一時ファイルは 100MB まで自由に利用できます。一時ファイルでは直接編成形式も可能で

```
$␣␣␣␣␣␣FILE␣␣␣␣10,,R
```

3 番目のオプションが R となります。

ジョブの投入/受取方法については第 5.2.3章をご覧下さい。

5.2.2 基本形用ジョブの作成

$NQS と $ENDJOB の間にシェルプログラムを書きます。UNIX を使われている利用者の方は非常に簡単に作成できると思われます。シェルプログラムの部分には SUPER-UX が提供しているコマンドが利用できます。使用するコマンドは小文字で指定して下さい。大文字を使用されると実行時, ' **XXX:Command not found.** ' が出力されエラーになります。

一般に SUPER-UX 上で FORTRAN77 プログラムを実行するためには, f77sx コマンドにより FORTRAN77 プログラムを翻訳/結合し, 実行形式ファイルを作成し, その実行形式ファイルを指定することにより, 利用者プログラムの実行を行います。なお, ACOS では利用者番号の下に

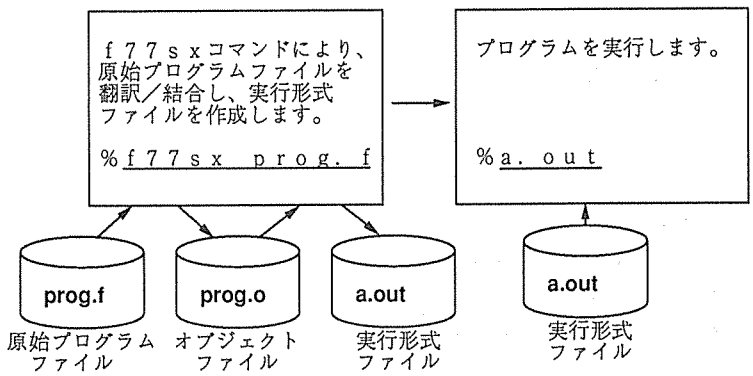

図 5.1: SUPER-UX 上での翻訳から実行までの過程

カタログあるいはファイルを作成しますが, SX[14]では登録番号の下にディレクトリあるいはファイルを作成します。また, 大文字と小文字は別とみなしますので, 同じファイル名でも大文字と小文字では別ファイルとなります。ご注意下さい。

翻訳する利用者のプログラムおよびデータは SX 上に必要です。SX を会話型で使用すればエディタなどで作成できますが, ファイルが ACOS にあれば簡単に転送できます。

```
stage␣W60000/CATA/SOURCE1␣/usr1/w60000a/source.f␣SOURCE
```

stage コマンドで ACOS のファイルを SX に転送します。

第 1 オプションで SX に転送する ACOS のファイルを指定します。ここでは利用者番号 W60000

[14]UNIX のファイル構造については「スーパコンピュータ利用の手引き (基本形編) をご覧下さい。

の下のカタログ CATA の下のファイル SOURCE1 を SX に転送しています。
2 番目のオプションとして SX 側のどのファイルに転送するかを指定します。ここでは登録番号 w60000a の下のファイル source.f に転送しています。ファイルが存在すれば上書きされ，なければ作成されます。SX のファイルを指定する場合，2 通りの方法があります。ここでは絶対パス名の方法で指定しています。絶対パス名の場合は必ず/usr1 から始め登録番号を書きファイルの指定となります。FORTRAN コンパイラで翻訳するときはソースファイルのサフィックスには.f が必要です。
3 番目のオプションとして転送するファイルの内容を指定します。ソースプログラムを転送するときは SOURCE オプションが必要です。

こうして転送したファイルを f77sx コマンドで翻訳/結合します。

```
f77sx␣/usr1/w60000a/source.f
```

この例は転送したファイル source.f をすべて既定値[15]の FORTRAN オプションで翻訳しています。すべて既定値で問題はありませんが，ただ一つプログラムの記述形式に注意する必要があります。既定値は f0

固定形式

が既定値です。自由形式 1 で書かれている場合は f1，自由形式 2 で書かれている場合は f2 を指定する必要があります。オプションは-をつけて指定します。
自由形式 2 で書かれているプログラムを翻訳します。

```
f77sx␣-f2␣/usr1/w60000a/src.f
```

重大レベル以上のエラーが発生すればオブジェクトプログラムは作成されませんが，エラーがなければ実行形式プログラムが

a.out

というファイルに作成されます。これを指定すれば実行できることになります。

実行形式プログラムを a.out 以外に作成したければ，o オプションを使用します。

```
f77sx␣-f2␣-o␣/usr1/w60000a/prog␣/usr1/w60000a/src.f
```

ここでは実行形式プログラムを登録番号の下のファイル prog に格納しています。

[15] 詳細は「スーパーコンピュータ利用の手引き」(基本形編) をご覧下さい。

センターで提供しているライブラリを使用する場合はライブラリのファイルを指定します。

```
f77sx␣-f2␣-o␣/usr1/w60000a/prog␣/usr1/w60000a/src.f␣/usr/lib/asl
```

ここでは科学技術計算ライブラリ ASL を割り当てています。指定するライブラリファイルは表5.2をご覧下さい。

表 5.2: ライブラリファイル名

ライブラリファイル名	ライブラリ名称
/usr/lib/imsl	IMSL ライブラリ
/usr/lib/asl	科学技術計算ライブラリ ASL/SX
/usr/lib/math	数値計算用ライブラリ MATHLIB/SX

利用者プログラムを実行する前に，外部装置識別子とファイルを割り当てる必要があります。システム標準ファイルは表5.3に示す外部装置識別子とそれぞれ事前に接続されています。

表 5.3: システム標準ファイル

外部装置識別子	システム標準ファイル
0	標準エラー出力ファイル
5	標準入力ファイル
6	標準出力ファイル

これらの外部装置は特にファイルに割り当てる必要はありませんが，その他の外部装置識別子を使用している場合は OPEN 文でファイルを割り当てない限り，カレントディレクトリ[16]の下に **fort.n** が作成され割り当てられます。n は外部装置識別子の番号です。これらの事前の接続を変更したい場合は，setenv コマンドを用いて変更します。ただし，setenv コマンドは C シェル用のコマンドです。

C シェルの場合

```
setenv␣F_FF09␣/usr1/w60000a/out09
```

この例は外部装置識別子 09 を w60000a の下のファイル out09 に割り当てています。

[16]登録番号がホームディレクトリとなり，変更しない限りホームディレクトリがカレントディレクトリとなります。詳細は基本形編をご覧下さい。

これで利用者プログラムを実行できますが，これだけではプログラム終了時に，プログラムの実行に関する詳細な情報が出力されません。このプログラム実行情報を出力するために，実行時オプション **F_PROGINF** が環境変数[17]として用意されています。プログラム実行前にこの環境変数に **YES** を設定して下さい。標準エラー出力ファイルに出力されます。

C シェルの場合

```
setenv␣F_PROGINF␣YES
```

ファイル入出力に関する情報を出力する **F_FILEINF**，書式つき記録が印字されるとき，その記録の先頭の 1 文字を行送り文字として扱う **F_VSPACING** 実行時オプションもあります。**YES** を設定して下さい。その他にも多数のオプション[18]があります。

C シェルの場合

```
setenv␣F_FILEINF␣YES
setenv␣F_VSPACING␣YES
```

SX のファイルを ACOS に逆転送するためには dstage コマンドを使用します。

```
dstage␣/usr1/w60000a/out09␣W60000/DATA09
```

SX の登録番号 w60000a の下のファイル out09 を ACOS 側の利用者番号 W60000 の下のファイル DATA09 に転送しています。

今までのコマンドをまとめて一つのバッチジョブを作成すれば，翻訳から実行までができます。

[17]利用者の作業環境を整えるために用意されている変数で，シェルが参照します。
[18]GUF 14-4「FORTRAN77/SX プログラミングの手引き」をご覧下さい。

例

```
$       JOB       ;A,W,,,,JPA4
$       NQS       TYPE=UX
# ..................................................................(a)
#
stage W60000/CATA/SOURCE1 /usr1/w60000a/source.f SOURCE ..........(b)
if ( $status != 0 ) then .........................................(c)
goto DONE ........................................................(d)
endif ............................................................(e)
f77sx -o obj /usr1/w60000a/source.f ..............................(f)
if ( $status != 0 ) then .........................................(g)
echo "source.f compile error" ..........................................(h)
goto DONE
endif ............................................................(i)
setenv F_FF10 /usr1/w60000a/da10 .....................................(j)
setenv F_PROGINF YES
setenv F_FILEINF YES
setenv F_VSPACING YES
obj ..............................................................(k)
dstage /usr1/w60000a/da10 W60000/DATA10 ..........................(l)
DONE:
if ( -f source.f ) then ..........................................(m)
rm source.f ......................................................(n)
endif
if ( -f obj ) then
rm obj
endif
if ( -f da10 ) then
rm da10
endif
$       ENDJOB
```

(a) 今まで説明しませんでしたが，#はシェルに対する注釈行となります。シェルにも数種類あり，ジョブ実行時にシェルを選択でき，シェルスクリプトの先頭に#があれば，C シェルが起動されます。したがって，この行以降，C シェル用の組み込みコマンドならびに SUPER-UX のコマンドを使用する事になります。センターでは C シェルが標準になっています。

(b) 利用者番号 W60000 の下のカタログ CATA の下のファイル SOURCE1 を，SX 上の登録番号 w60000a の下のファイル source.f に転送しています。ソースプログラムなので SOURCE オプションを指定しています。

(c) stage コマンドが正常に終了したかどうかチェックしています。正常に終了すれば status に 0 が入っています。

(d) 正常に終了していなければDONEへ飛びます。

(e) (c)のif文と対になります。

(f) source.fを翻訳/結合しています。実行形式プログラムを登録番号直下のファイルobjに格納しています。カレントディレクトリ[19]は変更しなければホームディレクトリとなります。

(g) コンパイルエラーがないかどうかチェックしています。エラーがあればstatusに0以外が入っています。

(h) エラーがあれば，標準出力ファイルに‘source.f compile error’を出力し，DONEに飛びます。

(i) (g)~(i)の文は一対でこれらの判定文がなければ，もしエラー発生時(k)のobjが存在すればそれが実行されますので注意が必要です。

(j) プログラム中の外部装置識別子10番のデータを登録番号w60000aの下のファイルda10に割り当てています。その他の環境設定も行っています。

(k) エラーがなければ実行形式プログラムの格納されたファイルを指定して実行します。

(l) SXのファイルに書き込んだデータをACOSに転送しています。

(m) source.fがあるかどうかチェックしています。

(n) ファイルがあれば削除しています。

5.2.3 簡易形/基本形用ジョブの投入

作成したジョブをカードインサブシステムを利用してSXに投入します。投入するジョブイメージが利用者番号W60000の下のファイルPRO1に作成されているとします。

システム選択レベルでCARDINサブシステムを呼び出します。

```
SYSTEM?CARD  N
```

次にRUNコマンドで作成したジョブが格納されているファイルを指定してジョブを投入します。

```
*RUN  /PRO1
```

システム側から

```
CRJE001 R CARD FORMAT,DISPOSITION?
```

の問い合わせがあるので，Card formatの問い合わせには行番号[20]があればS，無ければA，小文字を使用しているのであればL，基本形では必ず小文字を使用しているはずです。大文字だけの場合でもLを指定して構いません。DispositionにはJ又は省略応答を行います。

[19]「スーパーコンピュータ利用の手引き(基本形編)」をご覧下さい。
[20]行番号は取ってSXに渡さなければなりません。

CRJE001 R CARD FORMAT,DISPOSITION?S,L

次に

CRJE002 R TAB CHARACTER AND SETTING?

の問い合わせがあるので，タブ文字[21]を使用しているのであれば，そのタブと位置を入力します。使用していなければそのままキャリッジリターンキーを押します。例えば！[22]を使用しているのであれば

CRJE002 R TAB CHARACTER AND SETTING?!,8,16

システム側から受付番号が出力されます。ジョブの問い合わせ/検索にはこの受付番号を使用しますので忘れないようにして下さい。

CRJE350 I SNUMB 名 # 受付番号

ジョブの問い合わせにはJSTSあるいはJMONIコマンドを使用します。

*JSTS　受付番号

投入から問い合わせまでの例を示します。

例

```
SYSTEM?CARD  N ..........(a)
*RUN  /PRO1 ..........(b)
CRJE001 R CARD FORMAT,DISPOSITION?S,L ..........(c)
CRJE002 R TAB CHARACTER AND SETTING?!,8,16..........(d)
CRJE350 I SNUMB 名 # 0120T ..........(e)
*JSTS  0120T ..........(f)
```

(a) カードインサブシステムを呼び出しています。

(b) 利用者番号の下のファイルPRO1を投入しています。

(c) 行番号を削除し，小文字を使用している宣言を行っています。DISPOSITIONは省略しています。

(d) タブ文字として！を使用しています。1個目を8カラムに，2個目を16カラムに設定しています。

(e) システム側から受付番号が返されています。

(f) ジョブの状態を問い合わせています。

[21]空白を入れて桁合わせをしなくても済むように，特定の文字をある桁に合わせる機能。
[22]!,8,16とすれば$!JOB!;Aが$　　JOB　　;Aとなり，JOBが8カラム，;が16カラムから始まります。

5.2.4 出力結果

SX で計算が終了すれば ACOS の方に結果が転送されてきます。JSTS あるいは JMON コマンドで終了を確認し，JOUT コマンドで結果を検索できます。

- 簡易形

```
*JOUT  0110T
FUNCTION?LIST  ALL
ACTI# RC LINE HOLD CLASS
---  $$  --  --  I......(a)
001   00    2   YES   I(b)
001   74  498   YES   I(c)
003   06  214   YES   I(d)
FUNCTION?HOLD
```

- 基本形

```
*JOUT  0120T
FUNCTION?LIST  ALL
ACTI# RC LINE HOLD CLASS
---  $$  --  --  I......(a)
001   00  392   YES   I(b)
FUNCTION?HOLD
```

(a) 簡易形の場合，投入したジョブの JCL イメージ，指定されていれば変換された NQS イメージ，ジョブ実行情報が出力されています。基本形の場合は投入したジョブの JCL イメージおよびジョブ実行情報が出力されています。

(b) Warning: no access to tty; thus no job control in this shell.. が出力されています。これは利用者のバッチジョブを実行するためにシェルが起動されたが，tty モードなので使用できない機能がある旨の警告メッセージです。問題はありませんので無視して下さい。基本形の場合これ以外にソースプログラムの翻訳結果や，利用者の実行結果および指定すればプログラムの実行情報が出力されています。

(c) オプションが指定されていればソースプログラムのイメージが出力されています。

(d) 利用者プログラムの実行結果が出力されています。

5.2.5 簡易形から基本形への書き替え

簡易形ジョブでは利用者のプログラムの詳細な実行情報が出力されません[23]。どうしてもプログラムの詳細な実行情報が必要な方は基本形で実行していただくしか今のところ方法はありません。簡易形から基本形へ書き替えるための簡単な説明を行ないます。

(1) 簡易形ではソースプログラムおよびデータを$PRMFL 文あるいは$SELECTA 文で指定しますが，基本形では stage コマンドで ACOS のファイルを SX に転送します。また，WRITE(6… 以外の実行結果は SX のファイルに出力し，実行終了後，dstage コマンドで ACOS へ逆転送します。

(2) 簡易形では翻訳/結合を$FRT77 文で行いますが，基本形では f77sx コマンドで行います。

(3) 利用者プログラムの実行は簡易形では$GO 文ですが，基本形では timex コマンドで行います。

[23] 93 年 3 月 1 日現在。これが発行される頃には詳細な実行情報が出力されるようになっているかも知れません。

簡易形のジョブを基本形で記述した例を次に示します。

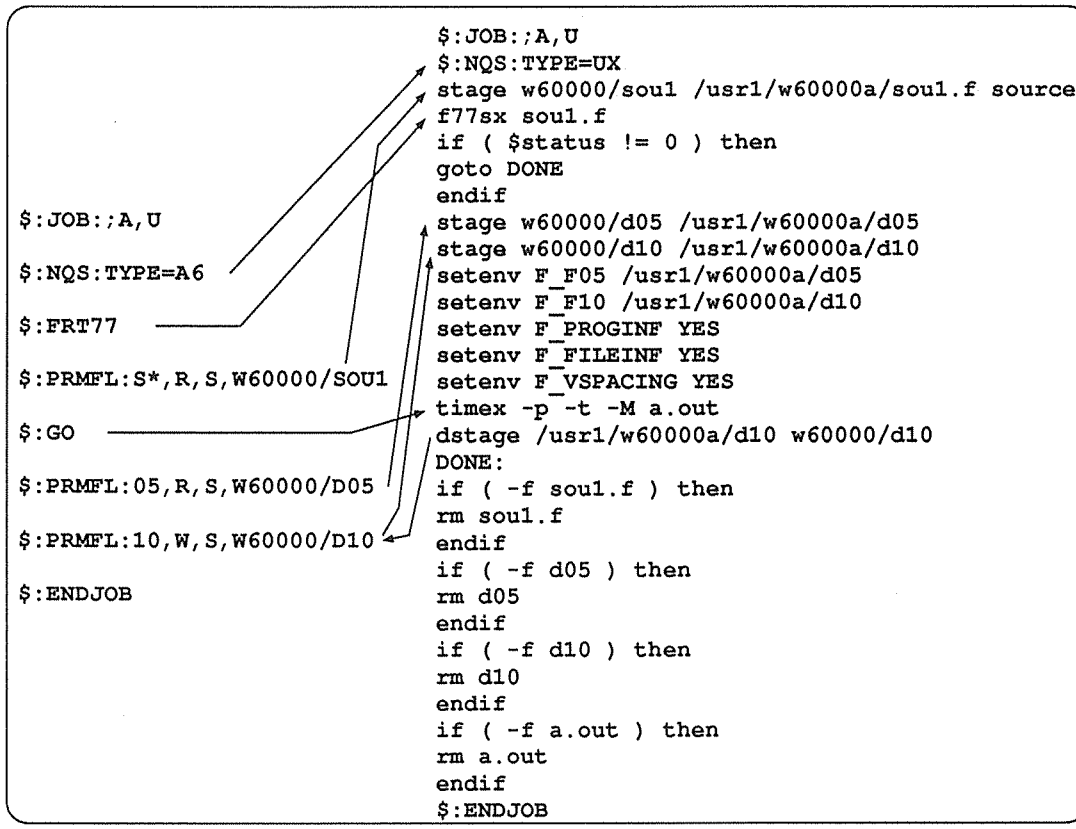

図 5.2: 簡易形から基本形へ

5.3 NQS 形用ジョブ

NQS 形を利用される方はすでに UNIX をご存知の方だと考えられますので，ここでは UNIX のコマンドの説明は行わず，NQS のオプション[24]と NQS のコマンドの説明を行います。

5.3.1 NQS 形用ジョブの作成

NQS では幾つかのコマンドの一括処理を依頼することをバッチリクエストといい，NQS に UNIX シェルスクリプトを投入することにより実現されます。このシェルスクリプトはコメント部分にオプションを記述できる点と処理の途中でインタラクティブな操作を必要とするようなシェルスクリプトを除き，UNIX で通常使用されているシェルスクリプトと何等代わるところはありません。また，このシェルスクリプトを解釈するシェルも自由に指定できますので，B-sh，Csh その他のシェル用のスクリプトでも構いません。また，オプションは書かないでリクエスト投入時に指定することもできます。

[24]オプションの詳細は「スーパーコンピュータ利用の手引き」(NQS 編) をご覧下さい。

ここでは，オプションをシェルスクリプトに埋め込んだ形で使用します。最初のシェルコマンドが現れる前のコメント部分に‘ @$ ’という文字列に続けてオプションを指定します。バッチリクエストの投入コマンドである qsub コマンドに指定できる全てのオプションが利用できますが，センターの運用上問題のあるオプションは無視されます。

```
#␣@$-q␣U
```

この例では，-q U が使用されているオプションで，バッチリクエストを投入するキュー名を指定しています。キューオプションは必須でキュー名には SX で実行するクラス U，V，W のいずれかのジョブクラス (表4.2参照) を指定します。

資源制限用オプションは数多くありますが，バッチリクエストの全プロセスに対するプロセス毎の CPU 時間の最大値を設定できます。設定しない場合は，各クラスの値が最大値となっています。

```
#␣@$-q␣W␣-lt␣"1:10:00"
```

上の例は W クラスで実行し，実行する CPU 時間を 1 時間 10 分に制限しています。

バッチリクエストの実行終了時にはジョブオカレンスレポート[25]が出力されます。このレポートを標準出力結果ファイルに出力するには jo オプション

```
#␣@$-jo
```

標準エラー出力用ファイルに出力するには je

```
#␣@$-je
```

オプションを使用します。

次に便利であると思われるオプションにメール関係の me オプションがあります。

```
#␣@$-me
```

この me オプションはリクエストの実行が終了したときにメールの発信を行うようにします。メールは投入もとのワークステーションの利用者に届きます[26]。ただし，異常終了した場合はオプションの有無に関わらずその内容を示すメールが投入者に送られます。

オプションは 1 行に複数書くこともできますし，行を分けて書くこともできますが，オプションの最後には必ずオプション列の終わりを示す‘ @$ ’を書きます。

```
#␣@$-q␣W␣-lt␣"1:10:00"
#␣@$-me␣-jo
#␣@$
```

[25] JOR:ジョブ実行情報

[26] NQS ジョブの投入はセンターの EWS で行いますので，EWS のホームディレクトリの下に.forward ファイルを作り転送先をいれておけばメールはそちらに転送されます。

このコメントの後にFORTRANプログラムを翻訳するf77sxコマンドや実行するコマンドを書きます。f77sxコマンドについては5.2.2「基本形用ジョブの作成」をご覧下さい。

例

```
#␣@$-q␣W␣-lt␣"1:10:00"
#␣@$-me␣-jo
#␣@$
cd␣~/f77
rcp␣ccews01:/usr1/w60000a/prog/main.f␣.
cd␣~/data
rcp␣ccews01:/usr1/w60000a/data/d1␣.
f77sx␣/usr1/w60000a/f77/main.f
if␣(␣$status␣!=␣0␣)␣then
    goto␣DONE
endif
setenv F_FILEINF YES ..... (a)
setenv F_PROGINF YES ..... (b)
setenv F_VSPACING YES ..... (c)

a.out␣<␣d1
DONE:
if␣(␣-f␣a.out␣)␣then
rm␣a.out
endif
if␣(␣-f␣d1␣)␣then
rm␣d1
endif
if␣(␣-f␣~/f77/main.f␣)␣then
rm␣~/f77/main.f
endif
```

例 1:FORTRAN 翻訳/実行

(a) 実行時のファイル入出力情報を標準エラー出力ファイルに出力します。

(b) プログラム実行に関する情報を標準エラー出力ファイルに出力します。

(c) 書式つき記録が印字されるとき，その記録の先頭の1文字を行送り文字として扱います。

標準出力および標準エラー出力の結果の受取については次をご覧下さい。

5.4 NQS形ジョブの投入/結果の受取

作成したジョブをNQSを利用してSXに投入します。前項の例1「FORTRAN翻訳/実行」のジョブがセンターのワークステーションccews01の登録番号w60000aの下のファイルpro1に作

成されているとします。

NQS のバッチリクエストの投入は qsub コマンドでスクリプトファイルを指定して行います。

```
ccews01%qsub␣pro1
```

ここではスクリプトファイル pro1 を指定しています。スクリプトファイルを指定しなかった場合は，標準入力ファイル stdin から読み込まれることになります。
システム側から受付が行われた旨のメッセージが出力されます。

```
Request nn.ccews01 submitted to queue: W.
```

nn.ccews01 がリクエスト ID と呼ばれるものでネットワークを通じてユニークなものです。nn には数字が入ります。‘ ccews01 ’は使用されるワークステーションにより異なります。最後には投入されたキュー名が表示されています。一旦ワークステーションのキューに登録され，その後 SX に転送されます。SX には ‘ ccsx3 ’というホスト名がつけられています。

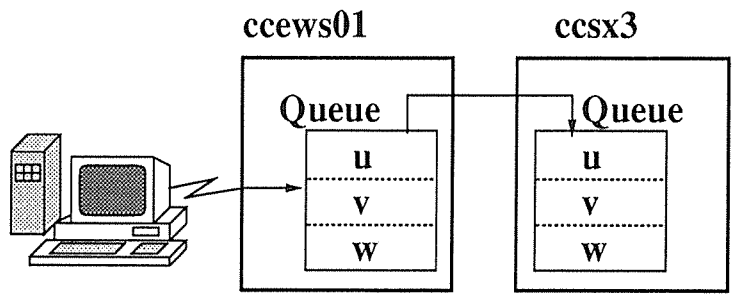

図 5.3: SX への投入

バッチリクエストの実行が終了すると二つの実行結果ファイルが得られます。標準出力用と標準エラー出力用の二つです。オプションを使用しなければ投入元のワークステーションのホームディレクトリの下に

標準出力結果ファイルは ‘ リクエスト名.o リクエスト連番 ’
標準エラー出力結果ファイルは ‘ リクエスト名.e リクエスト連番 ’

の形でファイルが作成されデータが書き込まれます。このときのリクエスト名はオプションでも指定できますが，指定していない場合はスクリプトファイル名がリクエスト名となります。リクエスト連番はリクエスト ID の前の数字の部分です。注意していただきたいのは，転送時，投入もとのワークステーションと何らかの理由で通信できない場合，SX の各利用者のホームディレクトリの下に書き込まれます。ジョブが終了したのに計算結果が返ってこないなどの場合，一度 SX のホームディレクトリの下を調べて下さい。会話型を利用すると簡単に調べることができます。もちろん，バッチジョブを作成しても調べることはできますが。

例

```
ccews01%qsub pro1
Request 12.ccews01 submitted to queue: W.
ccews01%
```

ここではファイル pro1 を投入し，リクエスト ID として ' 12.ccews01 ' が返されています。me オプションが使用されていますのでバッチリクエスト終了時にメールが ccews01 の利用者に届けられます。標準出力結果が **ccews01** のホームディレクトリの下のファイル pro1.o12 に，標準エラー出力結果が pro1.e12 に出力されます。

これらの出力結果を任意のファイルに出力する o と e オプション[27]があります。オプションの引数としてマシン名とパス名を書きます。

```
#␣@$-o␣ccews01:/usr1/w60000a/output/test1
#␣@$-e␣ccews01:/usr1/w60000a/error/test1
```

ここでは標準出力結果はセンターのワークステーション ccews01 の登録番号 w60000a の下のディレクトリ output の下のファイル test1 に，標準エラー出力結果は同じ登録番号の下のディレクトリ error の下の test1 ファイルに出力しています。

最後にワークステーション上のファイルをコピーして SX で実行する例を示します。ファイルのデータは次のようなものです。

[27] 2 月 15 日現在この機能は正常に働いていません。SX のホームディレクトリの下にファイルが作成されています。

```
#␣@$-q␣W␣-lt␣"1:10:00"␣-me␣-jo
#␣@$-o␣ccews01:/usr1/w60000a/output/test1
#␣@$-e␣ccews01:/usr1/w60000a/error/test1
#␣@$
cd␣~/f77
rcp␣ccews01:/usr1/w60000a/prog/main.f␣.
if␣(␣$status␣!=␣0␣)␣then
    echo␣"source␣file␣copy␣error."
    goto␣DONE
endif
cd␣~/data
rcp␣ccews01:/usr1/w60000a/data/d1␣.
if␣(␣$status␣!=␣0␣)␣then
    echo␣"data␣file␣copy␣error."
    goto␣DONE
endif
f77sx␣/usr1/w60000a/f77/main.f
if␣(␣$status␣!=␣0␣)␣then
    echo␣"compile␣error."
    goto␣DONE
endif
setenv␣F_FILEINF␣YES
setenv␣F_PROGINF␣YES
setenv␣F_VSPACING␣YES
a.out␣<␣d1
DONE:
if␣(␣-f␣d1␣)␣then
rm␣d1
endif
if␣(␣-f␣a.out␣)␣then
rm␣a.out
endif
if␣(-f␣/usr1/w60000a/f77/main.f␣)␣then
rm␣/usr1/w60000a/f77/main.f
endif
echo␣"Job␣Complete."
```

次の実行例は前記のジョブを ccews01 から投入しジョブの状態を確認しています。ジョブ終了後メールが届きますので，メールを確認しています。

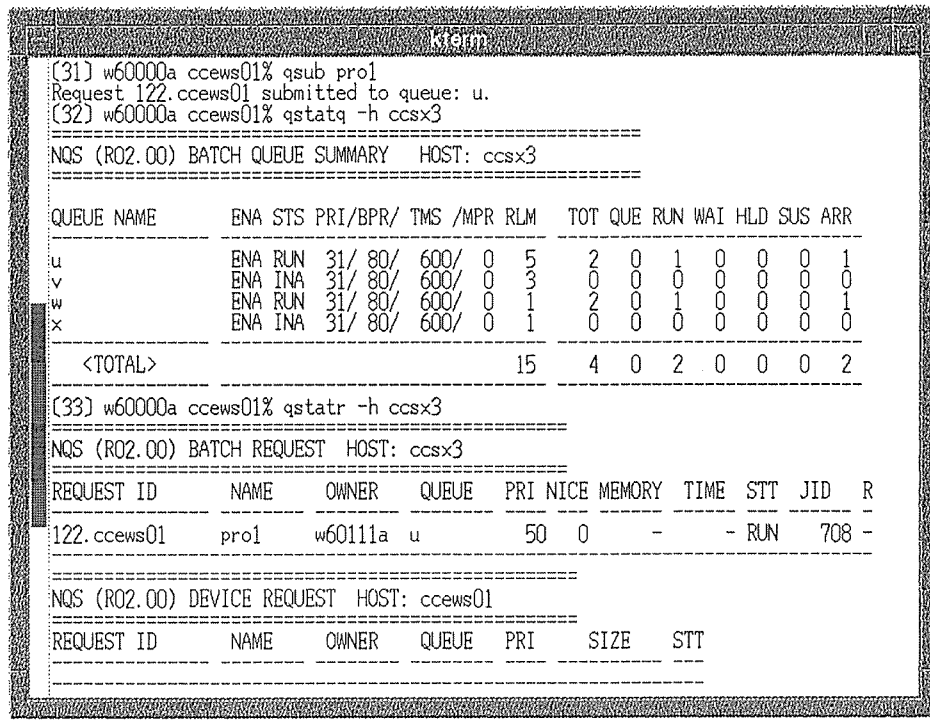

図 5.4: SX へのバッチリクエストの投入/確認

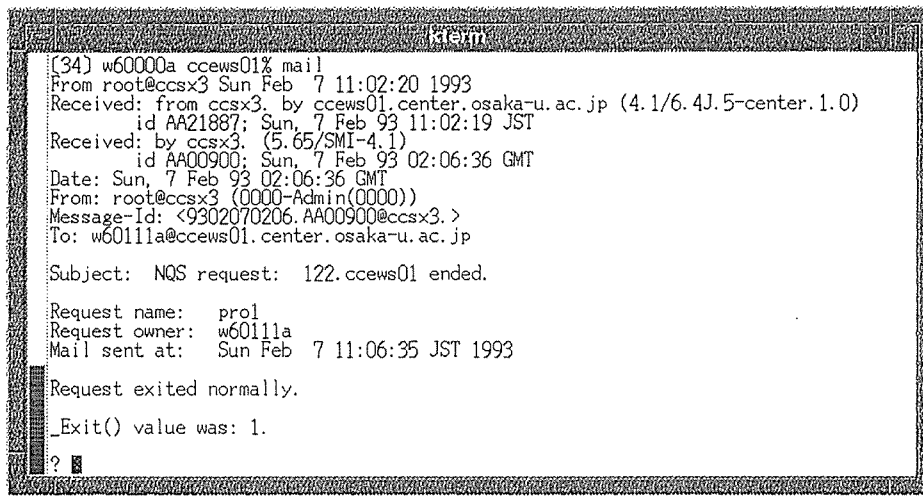

図 5.5: SX からのメールの確認

6　会話型直接利用形

ACOSあるいはワークステーションからSXへ直接接続し，会話型で使用することができます。もちろん，UNIXをよくしらなければ使えません。最近ではUNIXについて書かれた本がたくさん出版されています。UNIXを利用しようと考えられている方はそれらの本をご覧下さい。ここでは，基本形を使われている方が，SXのファイルの内容を確認したり，削除するための簡単な例を載せておきます。

会話型では利用できるリソースに制限があります。1つのコマンドでのCPU時間は10分，メモリサイズは40Mバイトとなっています。利用者のプログラムの実行もこの制限に当てはまります。これ以上必要な場合は，NQS形を利用しなければなりません。

ACOSからSXを使うためには，**TELNET**コマンドを使用し，計算機名に‘**CCSX3**’を指定します。

```
SYSTEM?TELNET ..... (a)
TELNET> ..... (b)
OPEN CCSX3 ..... (c)
SUPER-UX UNIX(ccsx3)

login: ..... (d)
w60000a ..... (e)
w60000a
Password:
xxxxxxxx ..... (f)
SUPER-UX Release 2.2 SX-3 ..... (g)
unix
   ⋮
% ..... (h)
```

(a) TELNETコマンドを入力しています。

(b) TELNETから入力促進記号が出力され，入力待ちとなります。

(c) SXへ接続するコマンドを入力しています。大文字で入力します。

(d) SXへ接続する利用者を問い合わせてきています。

(e) 登録番号を入力しています。小文字で入力します。

(f) パスワードを入力しています。

(g) SX側からメッセージが出力されています。

(h) 入力促進記号が出力され，コマンド入力待ちとなっています。

次にSXの中に入りましたので，lsコマンドでディレクトリおよびファイルの一覧を表示させてみます。SXと接続を切るときはlogoutコマンドを入力します。ACOSからSXへ入った場合は非常に使いにくい部分があります。この例はワークステーションから入った例を載せています。

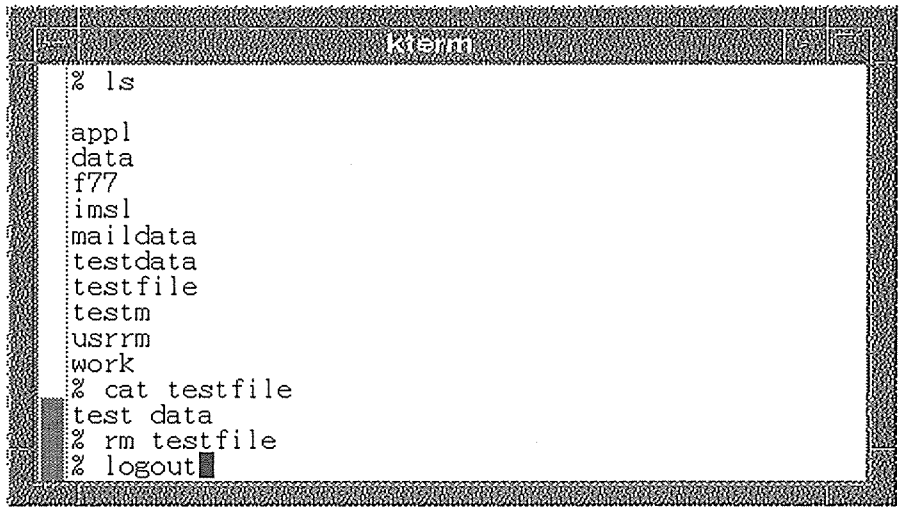

図 6.1: SX上でのlsコマンド

簡単な環境設定用ファイルを/usr1/guestの下に用意しています。各自コピーしてご利用下さい。ただし，SXにファイルを作成されますと，負担金が必要になります。

```
%cp␣~guest/.??*␣.
```

7 参考文献

次の説明書を参考にさせていただきました。

GUF11-3 SUPER-UX FORTRAN77/SX 言語説明書，NEC
GUF14-3 SUPER-UX FORTRAN77/SXプログラミングの手引，NEC
GUD11-3 SUPER-UX バッチ処理利用の手引，NEC
GUA11-3 SUPER-UX 利用者の手引，NEC

付録

A 利用形態の選択

いろいろな利用形態がありますので，どの利用形態を選択すればよいかの選択図を作成しました。ただし，ここにあげた条件が全てではありません。あくまでも目安として下さい。

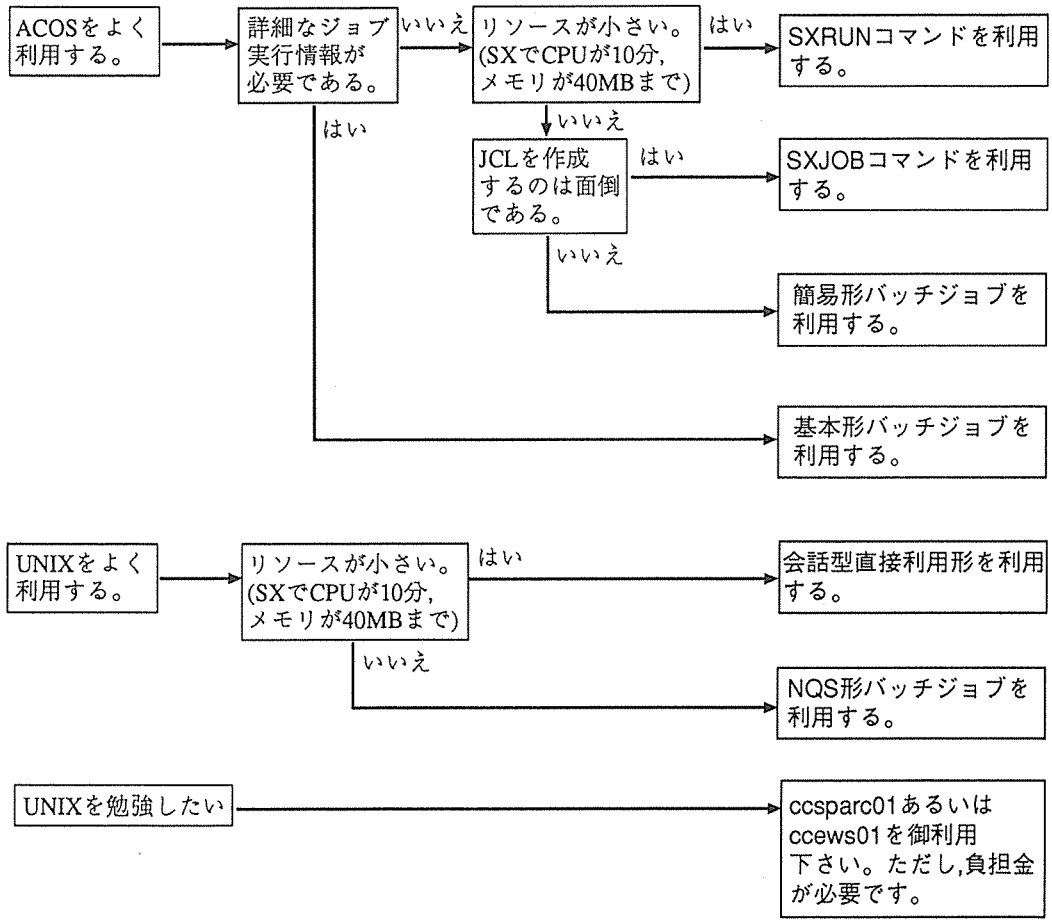
ACOSをよく利用する。
詳細なジョブ実行情報が必要である。
いいえ
リソースが小さい。(SXでCPUが10分,メモリが40MBまで)
はい
SXRUNコマンドを利用する。
いいえ
JCLを作成するのは面倒である。
はい
SXJOBコマンドを利用する。
いいえ
簡易形バッチジョブを利用する。
はい
基本形バッチジョブを利用する。
UNIXをよく利用する。
リソースが小さい。(SXでCPUが10分,メモリが40MBまで)
はい
会話型直接利用形を利用する。
いいえ
NQS形バッチジョブを利用する。
UNIXを勉強したい
ccsparc01あるいはccews01を御利用下さい。ただし,負担金が必要です。

図 a.1: 利用形態の選択