NAL TR-1410

ISSN 0389-4010
UDC 681.31

航空宇宙技術研究所報告

TECHNICAL REPORT OF NATIONAL AEROSPACE LABORATORY

TR-1410

数値風洞のジョブ処理におけるシステム構築

土 屋 雅 子

2000年8月

航 空 宇 宙 技 術 研 究 所

NATIONAL AEROSPACE LABORATORY

目　次

数値風洞のジョブ処理におけるシステム構築*

土　屋　雅　子*1

The System Design for Jobs in a Numerical Wind Tunnel*

Masako TSUCHIYA*1

ABSTRACT

The Numerical Wind Tunnel (NWT) is a distributed memory parallel computer for research and development in numerical simulation.
Effective utilization is a major consideration in the design of the NWT system.

We produced user interfaces in NWT in the same manner as in the existing system (Fujitsu FACOM VP-2100), comprising shell-scripts(NS-Shell) and command procedures(NS-Command). Numerous job-processing functions were included, resulting in a system that may be used easily and with confidence and delivering high overall performance. Experiments confirmed that all goals for system effectiveness were achieved.

In this paper we present concepts and detailed functions of system design for NWT jobs, along with their availability.

Keyword : system design, job, NWT, numerical simulation, parallel computer, UNIX, shell-script, command, user interface

概　　要

数値風洞（NWT）は、航技研の研究開発における数値シミュレーション処理を目的とした主記憶分散型並列計算機システムである。NWTの高度有効利用を実現するために、ジョブ処理におけるシステム構築を行った。ジョブ処理用のNSシェルとNSコマンドと呼称するユーザ・インターフェースを開発し、これらにNWTのジョブ処理に必要となる多くの機能を集約した。また、NSシェルとNSコマンドは航技研ユーザが利用に長けている従来システム（Fujitsu FACOM VP-2100）の形式とした。

実運用において、NSシェルとNSコマンドは有効に活用され、UNIXに不慣れなユーザの容易かつ確実なジョブ処理を実現している。その結果、NWTは所期の目標どおり、高度有効利用を可能にするシステム運用を実現した。

本稿では、NWTのジョブ処理におけるシステム構築とその有効性について報告する。

1. はじめに

平成5年2月、航技研に導入された科学技術計算用並列計算機システムの数値風洞[1),2)]（以降、NWTと略記）は大規模数値シミュレーションの計算エンジンとして絶大な力を発揮しており、先進的な航空宇宙技術の研究開発に重要な役割を果たしている。

平成8年1月にNWTのフロント・エンド・プロセッサ（以降、FEPと略記）が更新された。この更新前の旧FEPには二つのオペレーティングシステム（以降、OSと略記）が搭載されており、一つはNWTと同じUNIXシステム、他は従来から長年利用してきた汎用計算機システムのOSである富士通（株）製のMSPであった。旧FEPとNWTの運用システムでは、これらの二つの異種OSを連携し、MSPのユーザセッションからMSPビューのユーザ・インターフェース[3)]を介して数値シミュレーション処理を行ってきた。すなわち、各種のコマンド処理ならびにプログラム編集処理等の数値シミュレーション処理に必要

＊　平成12年4月13日受付　(received 13 April 2000)
＊1　計算科学研究部（Computational Science Division）

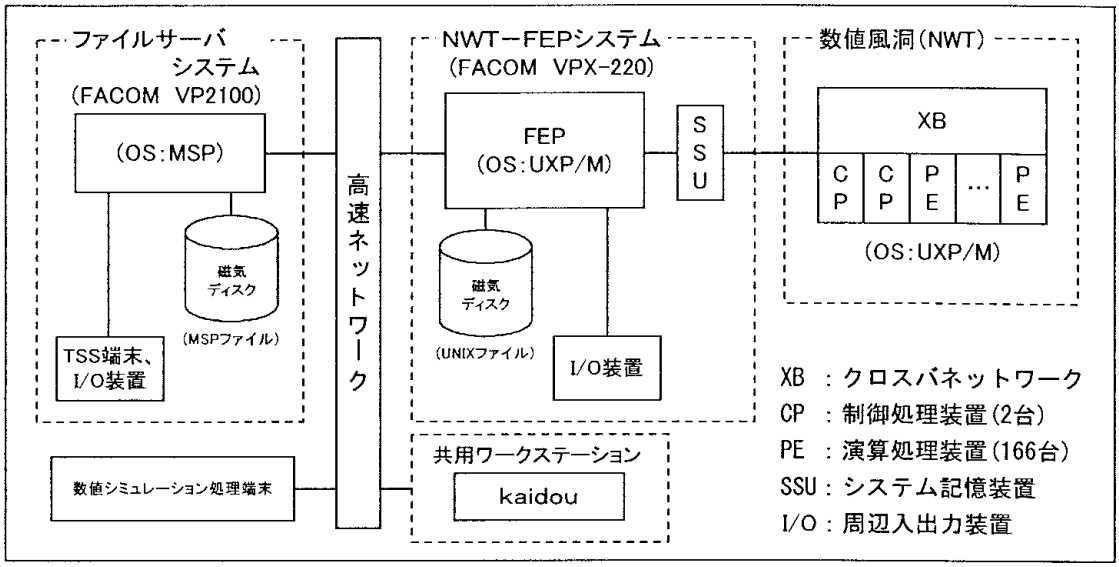

図2.1 中核部のハードウェア構成概念図

なマン・マシーン・インターフェース処理は航技研ユーザが従来から利用に長けていたMSPシステムを利用する運用とし、直接UNIXセッションからNWTを利用することは制限していた。

更新後の新FEPはUNIXだけを搭載するシステムに変更された。航技研の大型計算機システム利用における長い歴史の中で、新FEPはUNIXを初めてユーザ運用に供するシステムとなった。UNIXは豊富なネットワーク通信機能を持ち、オープン・インターフェースとして近年非常に普及しているOSであり、使い勝手や操作性の良さ、ならびに移植性の高さが定評なシステムである。また、ユーザ権限で実行できる機能が非常に多く、自由度の高いシステムでもある。一方、UNIXは不特定多数のユーザや時々刻々と大量に発生するジョブを運用管理するセンタマシーンのOSとしては、MSPのような従来の汎用計算機システムのOSに比べ運用管理機能が脆弱である。特に、UNIXには処理の投入から実行および実行結果の取り出しに至るまで、ジョブという処理単位で統合管理する機能がない。このため、航技研運用システムとして必須なジョブ処理機能を新たに付加し、ユーザが要求する大規模かつ長時間計算処理に対して、ジョブを構成する多数の各種処理フェーズをその要求どおりに正確な順序で、かつ順番に実行処理することを可能にした。

以上の観点から、NWTの超高速処理性能を十分に発揮し得、かつユーザ利用性の高い運用システムを実現するために、UNIXシステム上にNWTのジョブ処理におけるシステム構築を行った。このシステム構築の中から運用機能が多数用意されたが、この内、特に中心的な役割を果たすNSシェルとNSコマンドにNWTのジョブ処理に必要となる各種機能を集約した。なお、NSシェルとNSコマンドとは、これらについて端的に述べるならば、前者はジョブを記述する、後者はコマンドを記述するために必須なUNIXシステムのシェルスクリプト(注1)を生成するユーザインターフェースである。

実運用においては、NSシェルとNSコマンドは必要不可欠なユーザインターフェースとして有効に活用されている。この結果、NWTは常時、高効率なシステム利用状況を呈しており、また、ユーザには容易で、かつ確実なジョブ処理環境を供している。

本稿では、NSシェルとNSコマンドに集約したNWTジョブ処理におけるシステム構築とその有効性について報告する。

2. ハードウェア構成概要

NWTは要素計算機にベクトル計算機を配置する分散主記憶型の並列計算機システムである。図2.1にNWTの中核部のハードウェア構成概念図を示す。同図に基づいてNWTのハードウェア構成を概略する。NWTはVPX−220をFEPとして有機的に結合した複合計算機システムを構成している。その中核部は256MBメモリを実装した162台の要素計算機(PE)、1GBメモリを実装した4台のPE、OS処理のみを実行する制御処理装置(CP)2台、および、それらを相互接続するクロスバーネットワーク(XB)より構成される。NWTのハードウェアが有する超高速処理性能を十分に発揮するため、NWTには直接的には入出力装置を接続していない。CPはシステム記憶(SSU)を介

(注1) シェルスクリプト
複数のコマンド処理を順序どおりに実行させるための手順を記述したファイル。

してFEPに接続された磁気ディスク等の装置と入出力する。また、FEPにおける数値シミュレーションプログラムの開発処理やジョブ投入から実行結果の検索および取り出し等のジョブ操作に関するユーザのセッション処理において、FEPの負荷を軽減し、常時、良好な応答性を維持する目的から5台の共用ワークステーションを設置し、高速ネットワークに接続している。これらのワークステーションのセッション処理では、FEPのセッション処理の大部分を肩代わりしてジョブ処理に必要な操作を可能にしている。上記のハードウェア環境下でNWTは1台から166台までのPEを要求する小規模ならびに大規模数値シミュレーション処理のためのプログラム実行処理で常時混雑状況を呈している。

なお、NWTのOSはUNIXベースのUXP／Mである。また、NWTとFEPの緊密な連携をとるために親和性の観点からFEPのOSもUXP／Mを搭載している。一方、航技研では長年、富士通（株）製の汎用OSであるMSPを運用してきており、ユーザのファイルやノウ・ハウを含めた資産も膨大に存在するため、MSPビュー・ユーザ・インターフェースによるNWTのジョブ処理を実現している。これについては、平成6年2月末にファイルサーバシステムとして導入されていた汎用大型電子計算機システムFACOM　VP2100システムから、ネットワークを介してFEPと通信することにより、NWTジョブ処理の運用を実現している。なお、MSPビュー・ユーザ・インターフェースを利用するNWTのジョブ処理についての記述は別稿[3]に譲り、本稿ではUNIXシステム上に開設したユーザセッションから直接的に処理要求するジョブ処理におけるシステム構築について述べる。

3．数値シミュレーション処理の概要

数値シミュレーション処理で利用されるプログラムは、その殆どがFORTRAN言語で記述されている。数値シミュレーション処理はすなわちFORTRANプログラムの処理といえるが、これは翻訳処理、結合・編集処理および実行処理という三つの処理フェーズに区分できる。UNIXではこれらの処理フェーズをバッチリクエストというプロセスの集合を処理単位として管理している。本章では、FORTRANプログラムの処理概要とUNIX基本機能におけるバッチリクエストの処理フローの概要について示す。

3.1　FORTRANプログラムの処理概要

FORTRANプログラムの処理では各処理フェーズごとにユーザの開発したFORTRANソースプログラムのファイル以外に多数のファイルを入出力する。以下の各処理フェーズの概要説明に現れる各種入出力ファイルについては表3.1に示す。

(1) 翻訳処理

翻訳処理はメインプログラムやサブルーチン等、多数のプログラムで構成されるソースプログラムを言語処理プログラムのFORTRANコンパイラを使用してプログラム単位に翻訳処理を行い、プログラムの構文エラーを抽出する処理フェーズといえる。ユーザはコンパイラが標準出力する印刷出力ファイルから構文エラーの発生があるか否かを確認する。エラーがあった場合には、エラーをすべて除去し、ソースプログラムの修正のためのファイル編集作業を行う。プログラムの修正が終わると、再

表3.1　FORTRANプログラムの入出力ファイル

処理フェーズ	ファイルの種類	入出力モード
翻訳処理	FORTRANソースプログラム	入力
	標準印刷出力ファイル	出力
	オブジェクトモジュール	出力
結合・編集処理	オブジェクトモジュール	入力
	ユーザライブラリ	入出力
	システムライブラリ	入力
	ロードモジュール	入出力
	標準印刷出力ファイル	出力
数値シミュレーション実行処理	ロードモジュール	入力
	実行時標準入力ファイル	入力
	標準印刷出力ファイル	出力
	実行時使用入出力ファイル（大規模）	入出力
	実行時使用入出力ファイル（小規模）	入出力

度、翻訳処理を実行し、新たな構文エラーの発生を確認する。一般にプログラムの開発段階において、ユーザは翻訳処理を何回も繰り返し行い、ソースプログラムの構文エラーを完全に取り除くために計算機システムと向かい合い、完成されたプログラムへの非常に長い道のりを経なければならない。しかし、完成されたプログラムといえども、それは何百回、何千回も正常実行していながら、突然にプログラムエラーを発生することがある。このように、プログラムの開発はユーザにとって、非常に緻密さと複雑かつ困難な作業を強いる。以上のことから、計算機システムの運用では、プログラムの開発環境を最良のものとすることは非常に重要である。

翻訳処理におけるエラーがなくなると、その出力結果として一時ワークファイルまたはユーザ指定のファイルにバイナリ形式のオブジェクトモジュールが作成され、次の処理フェーズである結合・編集処理、さらに実行処理へと進むことができる。

なお、NWTでは、FEPでFORTRANプログラムの翻訳処理を実行するFORTRAN77クロスコンパイラとNWTで実行するFORTRAN90オウンコンパイラが提供されており、ユーザはそれらのコンパイラを任意に選択し、利用できる運用としている。

(2) 結合・編集処理

本処理は結合・編集処理プログラムのリンカーを使用して実行する処理フェーズである。結合・編集処理では、先行の翻訳処理が出力したオブジェクトモジュールを入力するとともに、ユーザプログラム内で自動呼び出しする各種のシステムライブラリならびにユーザライブラリ等を入力し、外部参照関係を解決するための結合・編集を行う。さらに、その実行結果として一時ワークファイルまたはユーザ指定のファイルに、一つのバイナリ実行形式のロードモジュールを作成する。

一般にユーザプログラムはメインプログラムと全サブルーチンを一体にした一個の実行可能プログラム形式のロードモジュールとして作成されることが多い。一方、結合・編集処理では、サブルーチン単位に一切の外部参照関係を未解決にしたライブラリ形式のロードモジュールを作成することも可能である。完成されたサブルーチンごとにライブラリ形式で作成すると、結合・編集処理フェーズの入力とし得るので、未完成の他のプログラムともに、ソースプログラムの状態から翻訳処理を繰り返し実行する必要がなくなり、効率的なジョブ処理が行える。この処理フェーズにおける処理の状況および正常またはエラー時のメッセージは標準印刷出力ファイルに示される。なお、以上のような各種、各様のプログラム形式入出力するには、これをシステムに正確に指示する結合・編集処理における手続きは非常に複雑かつ困難なものとなる。さらに、ユーザのプログラムのみならず、計算機メーカー提供のシステムライブラリや科学技術計算ライブラリ等、各種ライブラリを結合・編集するための多数の定義が必要である。

以上のとおり、数値シミュレーション処理における結合・編集処理では、オブジェクトモジュールならびに実行可能プログラム形式やライブラリ形式のロードモジュール等、各種状態のプログラムを結合・編集処理し、最終段階の出力結果として一個の実行可能プログラム形式のロードモジュール、すなわち数値シミュレーション実行可能プログラムを作成する。

(3) 実行処理

実行処理はユーザプログラムの本来の目的である数値シミュレーション実行処理を行う処理フェーズである。この処理フェーズでは、ロードモジュール形式のプログラムを入力し、さらに数値シミュレーション実行に必要となる各種の小規模および大規模データファイルを入出力しながら演算実行を行う。本処理フェーズにおける処理の状況および正常またはエラー時のメッセージは実行時標準印刷出力ファイルに示される。なお、ユーザのプログラムがロードモジュール形式となり完成プログラムの段階に近づいてもテストラン的な実行処理で各種のエラーが発生し、計算途中で終了する場合や期待した結果が得られない状況が多々発生する。ユーザはこの状況を打破するために本来のデバッグ計算処理やテスト計算処理を次から次ぎへと処理依頼する。このデバッグ計算やテスト計算処理段階においては、実行時の標準印刷出力ファイルに、エラー抽出のためにユーザがプログラムに挿入するwrite文による膨大なメッセージが出力されることが頻繁にある。このメッセージは実行処理に沿って時系列に順次出力されるので、プログラム開発のために必要不可欠な出力結果となる。これらの各処理フェーズで出力される印刷イメージの標準印刷出力ファイルは各処理フェーズにおける指針を与え、ユーザには非常に重要なものであるが、いったん確認が済むと、これらはゴミのファイルと化す。また、完成されたプログラムの処理に至っても、ユーザの指示がなければ、またこれらを出力不要にする指示が処理の手続き上で簡単に指示できなければ、これらを旧態依然として、システムは手続きに従って必ず出力することになる。これらは、ユーザの目にも触れられず、いつまでもシステムに有用なファイルと混在することになる。ファイル資源には限界があるので、ジョブ処理量が膨大になればなるほど、これらのファイルや各種処理フェーズで出力される一時ワークファイルは、システムが自動的に消去する仕組みが必要となる。また、プログラム開発過程で必要なこれら多種多様なファイルの出力についてはユーザが意識しないで

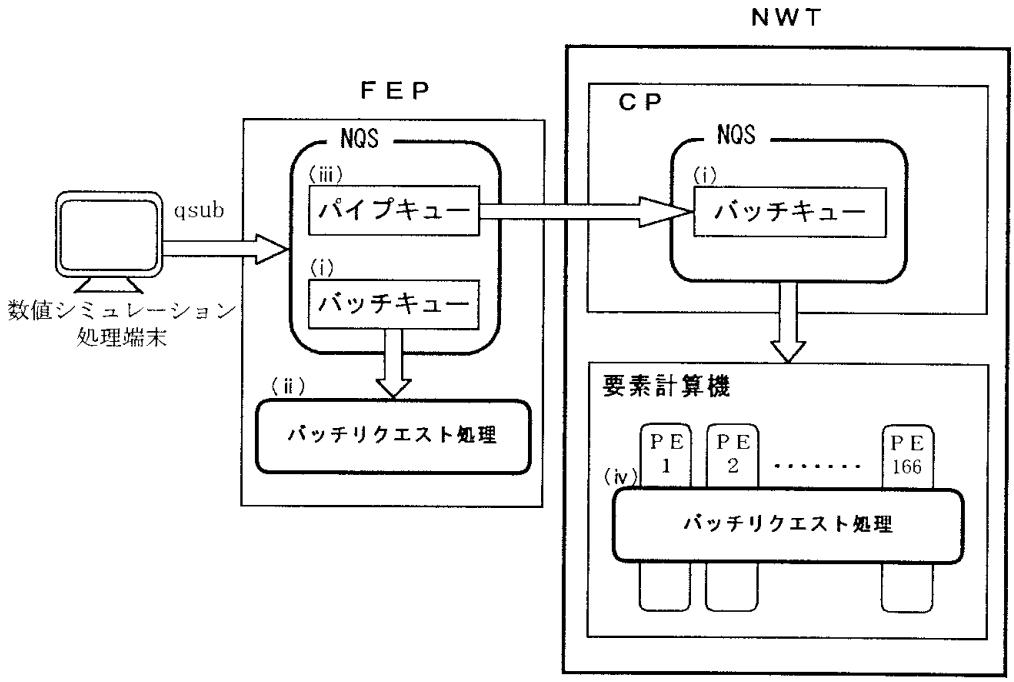

図3.1 バッチリクエストの処理フロー概念図

も、スペース不足等やユーザプログラムのエラーとは無関係な各種のエラー等が発生することがないようなジョブ処理の構築が必要である。

以上のとおり、ユーザがNWTにリクエストする数値シミュレーション処理の中からは、最終的な解を得るまでに多数のPEを要求する並列計算プログラムを実行し、短時間処理から非常に長大な処理時間を要する実行処理が多数発生する。

3.2 UNIX基本機能におけるバッチリクエストのフロー概要

UNIXシステムでは、上記で示したFORTRANプログラムの各処理フェーズをバッチリクエストという処理単位で管理することができる。バッチリクエストはコマンド列で構成されるシェルスクリプトで記述されており、これをqsubコマンドにより投入し、システムにバッチリクエスト処理の依頼を行う。図3.1にUNIXシステム基本機能におけるバッチリクエスト処理フローの概念図を示す。同図に基づいてUNIXシステムの基本的なバッチリクエスト処理フローの概要を以下に示す。

(1) FEPおよびCPのバッチキュー

(i) はNQS[4]、(注2)が管理するバッチキューであり、バッチリクエストを受付、実行させるためのキューである。

(ii) はFEPの処理装置で実行するバッチリクエストの処理である。UNIXのOSの実行管理下でバッチリクエストはバッチプロセスとして実行処理される。なお、端末のユーザセッションから投入される各種のコマンドは対話プロセスとして端末と直接通信を行いながらユーザと対話モードで実行処理されるが、バッチプロセスの処理は端末の接続とは無関係に実行処理される。

なお、クロスコンパイラを使用するFORTRANプログラムの翻訳、結合・編集処理のバッチリクエストはFEPのバッチリクエスト処理として定義できる。

FEPとCPにそれぞれのシステムのためのバッチキューがある。端末のセッションからqsubコマンドで投入されたバッチリクエストはシェルスクリプトの中に指定されたバッチキューにキューイングされる。NQSはバッチキューごとにCPUやメモリ等の各種資源使用量の制限値とバッチキュー優先権を定義できる。システムがバッチリクエスト起動可能な状態になると、NQSはバッチキュー優先権の高いキューより起動可能なバッチリクエストを検索して選び、これに処理装置を割当て、実行起動する。なお、FEPのバッチキューはFEP上のNQSがFEPで実行させるバッチリクエストを受付、実行を管理するバッチキューとして定義できる。また、CPのバッチキューはNWTに搭載される2台のCPの内、マスタCP上のNQSが管理するバッチキューであり、CPでバッチリクエストを受付、PEでバッチリクエストを実行させるためのリクエストを管理することが定義できる。

(2) FEPのバッチリクエスト処理

(注2) NQS
UNIXシステム上でユーザのリクエストを待ち行列で管理し、そのプロセスを実行起動するシステムプログラムである。

(3) パイプキュー

(iii) はFEPのNQSが管理するパイプキューである。パイプキューはバッチリクエストを受け取り、他の計算機システムにバッチリクエストを転送するためのキューである。このキューはFEPがバッチリクエストを受け取り、NWTへ転送するためのキューとなる。FEPのパイプキューから転送されたバッチリクエストはNWTのCPのバッチキューにキューイングすることが定義できる。

(4) NWT (PE) のバッチリクエスト処理

(iv) はNWTのPEで実行するバッチリクエストの処理である。CPのNQSが管理するバッチキューから送られたバッチリクエストは1台から166台のPEを割り当てられて並列プログラム処理を実行できる。なお、バッチリクエスト実行処理時のプロセス管理や入出力データ管理はNWTのUXP／M VPPシステムが制御する。なお、オウンコンパイラを使用するFORTRANプログラムの翻訳、結合・編集処理のバッチリクエストは1台のPEを割り当てられてNWTのバッチリクエストとして処理することが定義できる。

4. NWTジョブ処理におけるシステム構築

従来より、航技研の大型計算機システムでは、常時混雑するユーザの処理要求に対処するため、バッチジョブ処理を高効率に運用し得るシステム構築[5),6),7),8)]を行ってきた。この結果、計算機システムはどの世代のシステムでも、その時代における最新鋭機が導入されて、システムはいつも高稼働率で有効利用されてきた。また、航技研ユーザは計算機システムを利用すること自体を目的としている分けではなく、システムを研究の一道具として利用している。特に、新たなユーザインターフェースを習得するユーザ負担を軽くすることは、利用性の向上と考え、何時のシステムリプレースでもユーザインターフェースが変わる場合には、必ず旧インターフェースから新インターフェースへの容易な変換ツールを用意し、新システムを旧システムのインターフェースで利用することを可能にしてきた。この観点から、システム有効利用性の追求とともにユーザの利用性についても新たな提案や工夫を取り入れたシステム構築[9),10),11)]を行ってきた。本章では、平成8年の新FEP導入作業におけるNWTジョブ処理におけるシステム構築からNWTの高度有効利用を目標に開発された各種の機能について、その概要を述べる。なお、本章に示す各種機能により実現するジョブ処理における新規性について端的に示すと、表4.0のとおりである。

4.1 NSシェルとNSコマンドの開発について

NWTは航技研の計算機システムにおいて初めて運用するUNIXのOSである。長期間、MSPの利用に慣れてきたユーザにとって新たなUNIXの利用を強いられることは、大変な重荷を背負うこととなる。

一方、UNIXは従来よりユーザワークステーションのOSとして、少数ユーザがコマンドモードの会話型処理を主体として広く利用されてきた。従来の汎用計算機システムに比べUNIXでは、ユーザ権限で実行できるコマンドが多く、またコマンド処理自体がそのままユーザの処理要求に合致する標準コマンドを多数有している。また、先行処理したコマンドの結果を後続コマンド処理の入力にできる機能、すなわちパイプ機能については過去の航

表4.0 NWTジョブ処理における新規性

NWTジョブ処理における新規性		従来UNIXシステム利用方法との比較
NSシェルとNSコマンドの利用	従来ユーザインターフェースが継承できる。	従来ユーザインターフェースが継承できない。
	ユーザ個々に複雑なUNIXシェルを作成する必要がなく、ジョブの記述が非常に容易にできる。	ユーザ個々に複雑なUNIXシェルを作成する必要がある。
	数値シミュレーション処理を構成する複数のバッチリクエストを順番に、かつ順序どおりに実行することができる。	同時に複数のバッチリクエストを投入すると、処理の順番が保証されない。
	システム内の処理の混雑状況やターンアラウンドタイムの予測が容易である。	システム内の処理の混雑状況やターンアラウンドタイムの予測が容易でない。
	多数の処理要求においても実行結果の検索が容易である。	処理要求が多数になると実行結果との対応が困難となり、その検索が容易でない。
ジョブ名の発行	システム内の多数ジョブの中から、ジョブ名からその所有者と投入日の確認が容易である。	ジョブの所有者やその投入日は容易に確認できない。
同一ユーザ名のバッチリクエスト総数の制限	同一ユーザの処理要求によるシステムリソースの独占を回避できる。	同一ユーザの処理要求によりシステムリソースが独占状態となる可能性がある。
ジョブクラスの設定	ジョブクラス名により全バッチリクエストのシステム資源要求量が容易に判断できる。	バッチリクエストのシステム資源要求量が不明である。
NWTジョブ実行情報の出力	処理の経過が容易に確認できる。	処理の経過が容易に確認できない。
一時ワークファイルの自動消去	ファイル資源の無駄が省ける。	多数の不要ファイルによりシステム資源の枯渇を招く。

技研汎用計算機システムには持ち合わせない有効な機能である。このようなUNIXの特性を駆使すると、ユーザ自由度の高い、かつ操作性の高い、使い勝手の非常によいシステムとなる。

しかし、このUNIXの特徴は裏返すと、不特定多数ユーザが利用するシステムの運用管理の側面からは、システム資源の利用に無駄が生じ易い、ユーザ管理上に問題が発生し易いシステムと化す欠点となる。また、第3章で示したとおり、ユーザがリクエストする数値シミュレーション処理の翻訳処理、結合・編集処理および実行処理に対する処理要求の構成は千差万別であるが、従来の汎用計算機システムのようにUNIXにはジョブという単位で管理する概念がないので、UNIX基本機能では、このようなユーザのリクエストを要求どおりに正確な順序で、かつ順番に処理することができない。

以上の観点から、UNIXにおける数値シミュレーション処理のための一連のバッチリクエスト処理について、従来の汎用システムにおけるジョブという概念を取り入れ、NWTジョブという単位で効率よく管理するためのシステムを構築した。本システム構築において、UNIXシステム利用のためのユーザインターフェースとして開発したNSシェルおよびNSコマンドに、ユーザが容易かつ確実なNWTジョブ処理を実現するために必要となる各種機能を集約した。

先ず、NSシェルおよびNSコマンドの中で実現するNWTジョブの管理機能について以下に述べる。

4.1.1 ジョブの命名規約

NQSはシステムに投入されたバッチリクエスト毎に識別名として0～99999までの順序番号とホスト名からなるリクエスト識別子を発行する。このリクエスト識別子とは独立に、一回のサブミットコマンドで投入されたリクエストをNWTジョブとして管理するために、以下に示す命名規約に従ったジョブ名とバッチリクエスト名を設定する。

(1) ジョブ名の命名規約

投入されたNWTジョブごとに以下の形式の9文字からなるジョブ名を発行する。特に下5桁をジョブ番号とよぶ。

（ジョブ名の形式） x01 ＋ 25C3y
(a) (b) (c)

(a)ユーザ名

ユーザごとの3桁のユーザIDを設定する。また、航技研職員等を内部ユーザ、航技研が共同研究を行っている相手方の外部機関等のユーザを外部ユーザとして明確に区分して管理するために、ユーザIDごとにこれらのユーザ区分をシステムに定義した。このユーザ区分は航技研独自スケジューラのスケジューリング機能[11]の中で有効となり、内部ユーザが投入するNWTジョブは外部ユーザのそれより、ジョブのターン・アラウンド・タイム（ジョブ投入から実行処理終了までの経過時間）を短縮するスケジューリング調整に役立っている。

(b)ジョブ投入元の識別子

NWTジョブの投入方式により以下の識別子を設定する。

＋：NSコマンド（nsubおよびnsubo）投入の場合に付加する。

－：qsub投入の場合に付加する。

(c)ジョブ追番

ジョブの追番としてジョブ投入日時を示す5桁の番号（ddhmsの形式）を設定する。ddhmsの定義は、それぞれ以下のとおりである。

dd： ユーザリクエスト投入日付から、1～31までの2桁の整数を設定する。

hms：ユーザリクエスト投入時刻から、hは時刻、mは分、sは秒を意味し、それぞれ1桁で表す。なお、0から60までの数値を以下の文字で定義する。

（00～09：0～9、10～35：A～Z、36～59：a～x）

例えば、NWTジョブがx01＋25C3yというジョブ名を有する場合、ユーザ名はx01、投入日時は25日、12時03分58秒であることを意味する。

なお、qsub投入のジョブについてはジョブ番号の5桁の表示はユーザの任意としている。

(2) バッチリクエスト名

ジョブ名に続けてNWTジョブを構成する各バッチリクエストには、.（ピリオド）に続けて、以下の形式の5文字からなるバッチリクエスト名を付加する。

（バッチリクエスト名の形式）
x01＋25C3y . mm ／ nn
ジョブ名 (a) (b) (c) (d)

(a)常に.（ピリオド）を付加する。

(b)バッチリクエスト順序番号を意味する2桁の整数mmを設定する。ただし、バッチリクエストの種類がクロスコンパイル処理の場合には、mmの代わりに@xを、オウンコンパイル処理の場合には@oを設定する。

(c)常に／（スラッシュ）を付加する。

(d)投入された数値シミュレーション実行処理のバッチリクエスト総数を意味する2桁の整数nnを設定する。なお、翻訳処理のみのバッチリクエスト構成の場合には00を設定する。すなわち、クロスコンパイル処理のみのとき

バッチリクエスト名は@x／00、オウンコンパイル処理のみのときのバッチリクエスト名は@o／00を設定する。

4.1.2 同一ユーザ名のバッチリクエスト総数の制限機能

数値シミュレーション処理は解を得るまで非常に膨大な計算を行う必要がある。このため、システムが一人のユーザの独占状態とならないように、同一ユーザ名のバッチリクエスト総数の制限機能を設定する。本機能はシステムに同一ユーザのジョブ投入総数制限値をパラメータとして設定し、NWTジョブ投入時にそのユーザ名より、FEPおよびNWTに滞在する同一ユーザ名のジョブ総数を確認する。もし、ジョブ投入数が制限値を越える場合にはシステムは当該投入ジョブの受付を拒否し、実行処理を行わないこととする。また、この同一ユーザ名のバッチリクエスト投入制限値パラメータはNWTジョブの運用時間帯でも変更可能なパラメータとして定義する。

4.1.3 NWTジョブのジョブクラス設定

数値シミュレーション実行処理における各種資源の要求量の制限値を設定するため、また並列計算プログラムにおいてユーザが任意の並列度を定義でき、1PEから166PEの利用を定義するためのジョブクラスを設定した。また、ジョブクラスは利用PE数とPEの種類を明確に表すジョブクラス名を定義した。表4.1にNWTジョブのジョブクラスと各種資源使用制限値の設定を示す。NWTジョブのジョブクラスはユーザがNSシェルの中で指定するジョブクラスパラメータにより確定する。以下に表4.1の中の定義について示す。

(1) ジョブクラス名の設定

表4.1に示す4桁のジョブクラス名は数値シミュレーション実行処理においてユーザプログラムが要求するPEの種類と数を意味している。

ジョブクラスの上位1桁の英字は要求するPEの種類を意味し、搭載メモリが256MBのPEを要求する場合には英字uを、1GBのPEを要求する場合には英字lを指定する。

ジョブクラスの下位3桁の数字は要求するPEの数を意味し、要求するPE数を3桁で指定する。

ジョブクラス名は同表に示すとおり、256MBメモリのPEを要求するu001からu166までの166個のクラスと、1GBメモリのPEを要求するl001からl004までの4個のクラスを、それぞれ設定している。

(2) バッチリクエストの各種システム資源制限値

ジョブクラスごとにバッチリクエストの要求PE数、CPU時間および標準印刷出力量等のシステム資源に対して使用制限値を設定している。表4.1に示すとおり、各ジョブクラスとも、NWTジョブで使用する各PEごとのCPU最大使用時間は20,000秒（5時間33分20秒）とし、標準印刷出力量は同時にNWTで実行する同一ユーザの全バッチリクエストの出力合計値を最大3MBとして設定している。

なお、NWTジョブを構成する各バッチリクエストはバッチキューへキューイングするとき、ユーザが指定した各種のシステム資源要求量が確認され、バッチリクエ

表4.1 NWTジョブのジョブクラスと各種資源制限値

ジョブクラス名	システム資源使用制限値			
	PE数	メモリ容量	CPU時間	標準印刷出力量
Usual PE クラス (注1) u001 u002 ⋮ u166	 1 2 ⋮ 166	216MB × PE数	各PEごと 20,000秒	(注3) 3MB
Large PE クラス (注2) l001 ⋮ l004	 1 ⋮ 4	960MB × PE数		

(注1) Usual PE クラス ： 搭載メモリが256MBのPEを使用するジョブクラス。
(注2) Large PE クラス ： 搭載メモリが1GBのPEを使用するジョブクラス。
(注3) ユーザごとにNWTで同時に実行する全バッチリクエストの合計の出力量。

ストの要求がそれぞれの制限値を越えていなければ正常に受け付けられ、所定のバッチキューにキューイングされる。このとき、資源要求量のいずれかが制限値を越えているときには、バッチリクエストをリジェクトし、この旨のエラー発生をユーザにメールで通知する。

4.1.4 NWTジョブのバッチリクエスト処理待ちキューの設定

第3章で示したUNIXシステムのバッチキューおよびパイプキューについて、時々刻々と発生するNWTジョブを待ち行列として管理するためのキューとして定義する。自システムでバッチリクエストを処理するためのFEP用とNWT用バッチキュー、およびFEPからNWTへバッチリクエストを転送するためのパイプキューの設定について表4.2に示す。

同表に示すとおり、FEPバッチキューとして、ジョブクラスu001からu166のNWTジョブのためのjq01@aoiとジョブクラスl001からl004のNWTジョブのためのjq02@aoiを定義している。これらのバッチキューには、FEPにおける翻訳処理および結合・編集処理待ちのバッチリクエストをキューイングする。

NWTバッチキューとして、ジョブクラスu001からu166のNWTジョブのためのjq01@nwtcp1とジョブクラスl001からl004のNWTジョブのためのjq02@nwtcp1を定義している。これらのバッチキューには、NWTにおける数値シミュレーション実行処理待ちのバッチリクエストとオウンコンパイラとリンカーによる翻訳処理および結合・編集処理待ちのバッチリクエストをキューイングする。

FEPパイプキューには、ジョブクラスu001からu166のNWTジョブのためのnjq01@aoiとl001からl004のNWTジョブのためのnjq02@aoiを定義している。これらのパイプキューはFEPからNWTへ転送されるバッチリクエストが一時的に通過するキューである。

4.1.5 NWTジョブの出力情報

NWTジョブの処理では、第3章に示すとおりユーザプログラムの翻訳処理、結合・編集処理ならびに数値シミュレーション実行処理の結果として、本来のバッチリクエスト実行結果やシステムが標準的に出力する印刷イメージの出力結果等がファイルに多数出力される。

一方、ジョブの投入からジョブがシステムを離脱するまでの処理過程で、ユーザが記述するジョブの手続きには現れない各種のOSプロセスからの出力情報も多数ある。これらの情報の中には、ユーザが一刻も早く実行結果を得たいという気持ちから、処理の経過を見守る上で特に必要となる情報、すなわちジョブの受付メッセージ、各バッチリクエスト実行開始／終了メッセージ等が含まれる。これらの詳細なジョブ処理状況を示す情報をNWTジョブごとにログ情報として管理し、ユーザが時々刻々と経過していくNWTジョブの処理過程を容易に確認できるようにした。さらに、特に重要な情報はユーザのセッション画面にメッセージ出力し、またバッチリクエストの実行開始／終了メッセージについては当該ユーザにメールで通知することとした。

また、第3.1節で示したとおり、各種のバッチリクエスト処理における出力結果として、オブジェクトモジュールやロードモジュール等のように次の処理フェーズには必要であるが、ジョブが終了すれば用済みとなる、ユーザが陽に保存を指示しない一時ワークファイルが多数出力される。このため、時々刻々と終了していく多数のNWTジョブから発生する各種の出力ファイルは膨大な量となり、保存領域として必要なファイルスペース量は際限がない。

以上の観点から、NWTジョブから出力される各種のファイルのうち、ユーザがジョブ終了後に確認すれば、消去可能なファイルを一時ワークファイルとして、その出力元NWTジョブを明確に示す特定の名前のファイル

表4.2 NWTジョブのバッチキュー

ジョブクラス名	FEPバッチキュー	FEPパイプキュー	NWTバッチキュー
Usual PE クラス u001 u002 ⋮ u166	jq01@aoi	njq01@aoi	jq01@nwtcp1
Large PE クラス 1001 ⋮ 1004	jq02@aoi	njq02@aoi	jq02@nwtcp1

に出力することとした。この内、ジョブ終了後には不要となる一時ワークファイルは終了時に消去することとした。また、ユーザが確認する必要のある一時ワークファイルは期限付きファイルとして定義し、一定期間参照されない場合には、これらをシステムが自動的に消去する仕組みとした。表4.3にNWTジョブの出力情報の種類とその出力先および出力ファイル名を示す。

4.2 NSシェルの機能概要

通常、ジョブの記述にはシステムプログラムや表3.1に示す様な各種入出力ファイル等、ユーザには明解でない実行定義文や多数のファイル定義文が必要である。このため、ジョブの記述を完成させるには、多大な労力と非常に緻密な作業を要する。このようなジョブの記述におけるユーザ負担を軽減し、NWTジョブをユーザが容易かつ確実に記述することを可能にするために、さらに、記述上の誤りによる無駄な処理を省く等、システムの高度有効利用を達成し得るジョブ処理を実現するためにNSシェルを開発した。

NSシェルは実行プログラム名や入出力ファイル定義文等のジョブを記述するUNIXのqsubのシェルスクリプトを簡略化およびマクロ化した定義文である。NSシェルでは大方の処理に共通な手続きをシェル展開時に自動展開することにより、複雑かつ多数の定義を削減でき、ユーザジョブに固有の定義についてのみ、パラメータで簡単に指定できる仕組みとしている。また、NSシェルはバッチリクエスト実行時のCPU使用時間、メモリおよびファイル等のシステム資源使用量についてはシステム標準値やセンタが決める省略値を有し、ユーザがそのパラメータの指定を省略すると、所定の標準値または省略値が設定されるようになっている。この標準値および省略値にはシステムが効率的に運用管理できるようにチューニングされた値を定義している。したがって、NSシェルを活用すれば、目的とするジョブの記述が非常に容易に、かつ確実となり、さらにシステム資源の無駄等も省けるので、ユーザのみならずシステム運用管理上も利点が大きい。また、NSシェルの記述形式は航技研ユーザが非常に慣れており、利用に長けている従来システムのMSPの形式を取り入れ、UNIXに慣れていないユーザでも簡単に利用できるものとした。以下にNWTジョブ処理用に開発されたNSシェルの機能概要を示す。詳細な記述形式および機能については参考の表中に表す。なお、各項に付記する参考の表については文末に一括して配置する。

(1) njob文

NWTジョブの先頭を示す指定必須のNSシェルである。njob文では、プログラムの並列度に対応するジョブクラスおよびバッチリクエストのCPU打ち切り制限値等が記述できる。表4.4にnjob文の記述形式および機能詳細を示す。

(2) nfortc文

NWTジョブのFORTRANソースプログラムの翻訳処理を実行することを指示するNSシェルである。nfortc文

表4.3 NWTジョブの出力情報

	NWTジョブ出力情報の種類	出力先および出力ファイル名
(1)	ジョブの実行処理の状況を示すジョブログ情報	/nwjtmp/グループ名/ユーザ名/log
(2)	NQS 受付メッセージ	nsub投入セッションの標準出力
(3)	NSシェル展開処理時のメッセージ	nsub投入セッションの標準出力
(4)	バッチリクエストの実行開始/終了メッセージ	NSシェルで指示した場合、ユーザにメールで通知
(5)	実行時標準出力の書戻し完了メッセージ	/nwjtmp/グループ名/ユーザ名/ジョブ名/log
(6)	バッチリクエストのｑｓｕｂシェルスクリプト	/nwjtmp/グループ名/ユーザ名/ジョブ名/shell##
(7)	翻訳処理時の標準印刷出力	/nwjtmp/グループ名/ユーザ名/ジョブ名/fortc##
(8)	結合・編集処理時の標準印刷出力	/nwjtmp/グループ名/ユーザ名/ジョブ名/lied##
(9)	ユーザライブラリ	/nwjtmp/グループ名/ユーザ名/ジョブ名/ulib
(10)	実行時の標準印刷出力および標準エラー出力	/nwjtmp/グループ名/ユーザ名/ジョブ名/go##
(11)	オブジェクトモジュール	/nwjtmp/グループ名/ユーザ名/ジョブ名/a.o
(12)	ロードモジュール	ユーザ指定のファイル名または省略時（以下） /nwjtmp/グループ名/ユーザ名/ジョブ名/a.out
(13)	翻訳処理時のシステム一時ワークファイル	/nwjtmp/グループ名/ユーザ名/ジョブ名/*.s

（注）##に示すファイルはバッチリクエスごとに出力され、バッチリクエスト番号または記号［01-99, @X, @O］を付加する。）

は複数文記述可能とし、FORTRANコンパイラオプションおよび入力するソースプログラムのファイル名等が記述できる。また、ソースプログラムはヒアドキュメントとして、nfortc文に続けて記述可能である。また、翻訳処理が正常終了するとオブジェクトモジュールを一時ワークファイルに出力することを指示する。表4.5にnfortc文の記述形式および機能詳細について示す。

(3) nlied文

nfortc文による先行の翻訳処理が出力するオブジェクトモジュールを入力し、結合・編集処理することを指示するNSシェルである。nlied文はnfortc文の次に現れ1文のみ記述可能とし、結合・編集処理時のオプションと実行結果としてのロードモジュールを作成、保存する場合のファイル名が指定できる。また、科学技術計算ライブラリおよびセンタライブラリ等のライブラリファイルの参照がシステム標準指定として定義されている。表4.6にnlied文の記述形式および機能詳細について示す。

(4) ngo文

nlied文による結合・編集処理で作成、保存されたロードモジュールを入力し、NWTで数値シミュレーションプログラムの実行処理を指示するNSシェルである。ngo文では実行時のオプション、CPU打切り時間および既存の実行プログラムとしてユーザ保存ファイル名等の指定が可能であるとともに、実行時使用標準入力ファイルとしてユーザ保存ファイルが指定可能である。また、カードイメージのデータ入力をヒアドキュメントとしてNGO文に続けて定義できる。表4.7にngo文の記述形式および機能詳細について示す。

(5) nusdkr文

NWTにおける数値シミュレーション実行時使用入出力ファイルの内、入力ファイルの使用を指示するNSシェルである。表4.8にnusdkr文の記述形式および機能詳細について示す。

(6) nusdkrw文

NWTにおける数値シミュレーション実行時使用入出力ファイルの内、入力と出力をともに行うファイルの使用を指示するNSシェルである。表4.9にnusdkrw文の記述形式および機能詳細について示す。

(7) nusdkw文

NWTにおける数値シミュレーション実行時使用入出力ファイルの内、出力ファイルの使用を指示するNSシェルである。表4.10にnusdkw文の記述形式および機能詳細について示す。

(8) nxy文

NWTにおける数値シミュレーション実行時使用入出力ファイルの内、静的図データファイルの出力を指示するNSシェルである。FEPのセッション画面でnxy文による出力ファイルをxyプロッタイメージで表示し、数値シミュレーション実行処理の検証を行うことができる。表4.11にnxy文の記述形式および機能詳細について示す。

(9) nulib文

NWTジョブで使用する自動呼び出し形式のユーザライブラリを作成することを指示するNSシェルである。nulib文はnjob文の次に現れ、1文のみ記述可能とし、FORTRANコンパイラオプションおよび入力するソースプログラムのファイル名等が記述できる。表4.12にnulib文の記述形式および機能詳細について示す。

4.3 NSコマンドの機能概要

NSシェルを利用するNWTジョブ処理に関連するユーザの端末入力を支援するためのコマンドを作成し、これをUNIX基本コマンドと区別するためにNSコマンドと呼称する。NSコマンドはNSシェルと同様に、複雑で多数の定義を要するコマンド処理手続きについて、大方のコマンド処理に共通なことは入力を省略し得、コマンド展開時にこれを自動展開するので、ユーザのコマンド実行に必要な固有の定義についてのみ、コマンドのオプションで簡単に指定できる仕組みになっている。また、コマンド実行に必要な各種のシステム資源使用量についてもNSシェルと同様、運用効率を高めるようなシステム標準値や省略値を定義している。したがって、NSコマンドを活用すると、会話型処理におけるNWTジョブの投入や実行結果確認等のジョブ操作を容易かつ効率的に行える。なお、NSコマンドはFEPのセッション処理における負荷を軽減するため、リモートホストから投入可能なコマンドについてはkaidouのセッションからも投入できる仕様としている。以下にNSコマンドの入力形式および機能概要等について述べる。なお、各項に付記する参考の表については文末に一括して記述する。

(1) actjobコマンド

FEPおよびkaidouのセッションからNWTジョブの実行状況の表示を指示するコマンドであるactjobコマンドでは、FEPおよびNWTで実行中の全NWTジョブについて割当PE台数、CPU使用時間ならびに経過時間等の詳細なジョブ実行状況を表示する。表4.13にactjobコマンドの入力形式および機能概要について示す。

(2) allocコマンド

FEPのセッションからNWTジョブの実行時に使用する大規模データファイルを予め割当することを指示するコマンドである。表4.14にallocコマンドの入力形式および機能概要について示す。

(3) f77nwtコマンド

FEPのセッションからFORTRANソースプログラムを翻訳処理、結合・編集処理および実行処理までを実行す

ることを指示するコマンドである。f77nwtコマンドでは、FORTRAN77 コンパイラオプションおよび入力するソースプログラムのファイル名ならびに結合・編集処理時のオプション等が指定できる。また、科学技術計算ライブラリおよびセンタライブラリ等の数値シミュレーション処理に必要なライブラリファイルの参照がシステム標準設定として定義されている。表4.15にf77nwtコマンドの入力形式および機能概要について示す。

（4）ncan コマンド

ncan コマンドでは、FEP および NWT で実行待ち、実行中等、各種の状態における NWT ジョブのキャンセルを指示できる。表4.16にncanコマンドの入力形式および機能概要について示す。

（5）nlog コマンド

FEPおよびkaidouのセッションから実行結果を検索するコマンドである。nlogコマンドでは、NWTジョブ名を指定すると、当該ジョブの各バッチリクエストで出力した印刷イメージの標準出力結果について検索できる。表4.17にncanコマンドの入力形式および機能概要について示す。

（6）nquota コマンド

FEPおよびkaidouのセッションからファイルの使用状況を画面に表示することを指示するコマンドである。コマンドでは、NWTジョブで利用するユーザ貸出しファイルの割当量、使用量ならびにファイル数をファイルシステム単位に出力する。表4.18にnquotaコマンドの入力形式および機能概要について示す。

（7）ns コマンド

FEP および kaidou のセッションからユーザの NWT ジョブ処理状況表示を指示するコマンドである。nsコマンドでは、当該ユーザの全 NWT ジョブについて実行待ち、実行中および実行結果出力待ち状態であるかを表示する。表4.19にnsコマンドの入力形式および機能概要について示す。

（8）nsub コマンド

FEP および kaidou のセッションから、NS シェルで記述されたジョブストリームを格納しているユーザ保存ファイル名を指定して NWT ジョブの投入を指示するコマンドである。nsub コマンドでは翻訳処理および結合・編集処理はFEPでクロスコンパイラを使用して実行される。表4.20にnsubコマンドの入力形式および機能概要について示す。

（9）nsubo コマンド

FEP および kaidou のセッションから、NS シェルで記述されたジョブストリームを格納しているユーザ保存ファイル名を指定して NWT ジョブの投入を指示するコマンドである。nsubo コマンドでは、翻訳処理および結合・編集処理についても NWT でオウンコンパイラを使用して実行される。表4.21に nsubo コマンドの入力形式および機能概要について示す。

（10）nulib コマンド

FEP のセッションから NWT ジョブで使用する自動呼び出し形式のユーザライブラリを作成することを指示するコマンドである。nulib コマンドでは、FEP のクロスコンパイラを使用し、翻訳処理および結合・編集処理を実行する。表 4.22 に nulib コマンドの入力形式および機能概要について示す。

（11）rlse コマンド

FEP セッションから NWT ジョブの実行時に使用される大規模データ用vflファイルの未使用領域を開放することを指示するコマンドである。表 4.23 に rlse コマンドの入力形式および機能概要について示す。

（12）vflrm コマンド

FEP セッションから NWT ジョブの実行時に使用される大規模データ用vflファイルを削除することを指示するコマンドである。表 4.24 に vflrm コマンドの入力形式および機能概要について示す。

（13）vfluse コマンド

FEP セッションから NWT ジョブの実行時に使用される大規模データ用vflファイルの使用状況を確認することを指示するコマンドである。vfluseコマンドでは、コマンド投入時に指定ファイルが NWT で実行中のバッチリクエストが使用しているか否かを示す。表 4.25 に vfluse コマンドの入力形式および機能概要について示す。

（14）waitjob コマンド

FEP および kaidou のセッションから実行待ち状態の NWT ジョブを表示するコマンドである。waitjob コマンドでは、NWTにおける数値シミュレーション実行処理を起動待ちしている全ユーザジョブについて要求PE台数、要求CPU使用時間ならびに起動優先権等、詳細なジョブ実行待ち状況を表示する。表4.26はwaitjobコマンドの入力形式を示す。

5. NWT ジョブ処理の流れ

本章では、NWT のジョブ処理用に開発された NS シェルやNSコマンド等を利用し、数値シミュレーション処理端末から投入された NWT ジョブの処理の流れについて述べる。図5.1はNWTジョブ処理の流れの概念図を示す。同図に基づいて、本システム構築により実現した NWT ジョブ処理の流れの概要を以下に示す。

5.1 NWT ジョブの投入

NWTジョブの処理は各種の数値シミュレーション処理端末で開設する FEP および共用ワークステーション（以

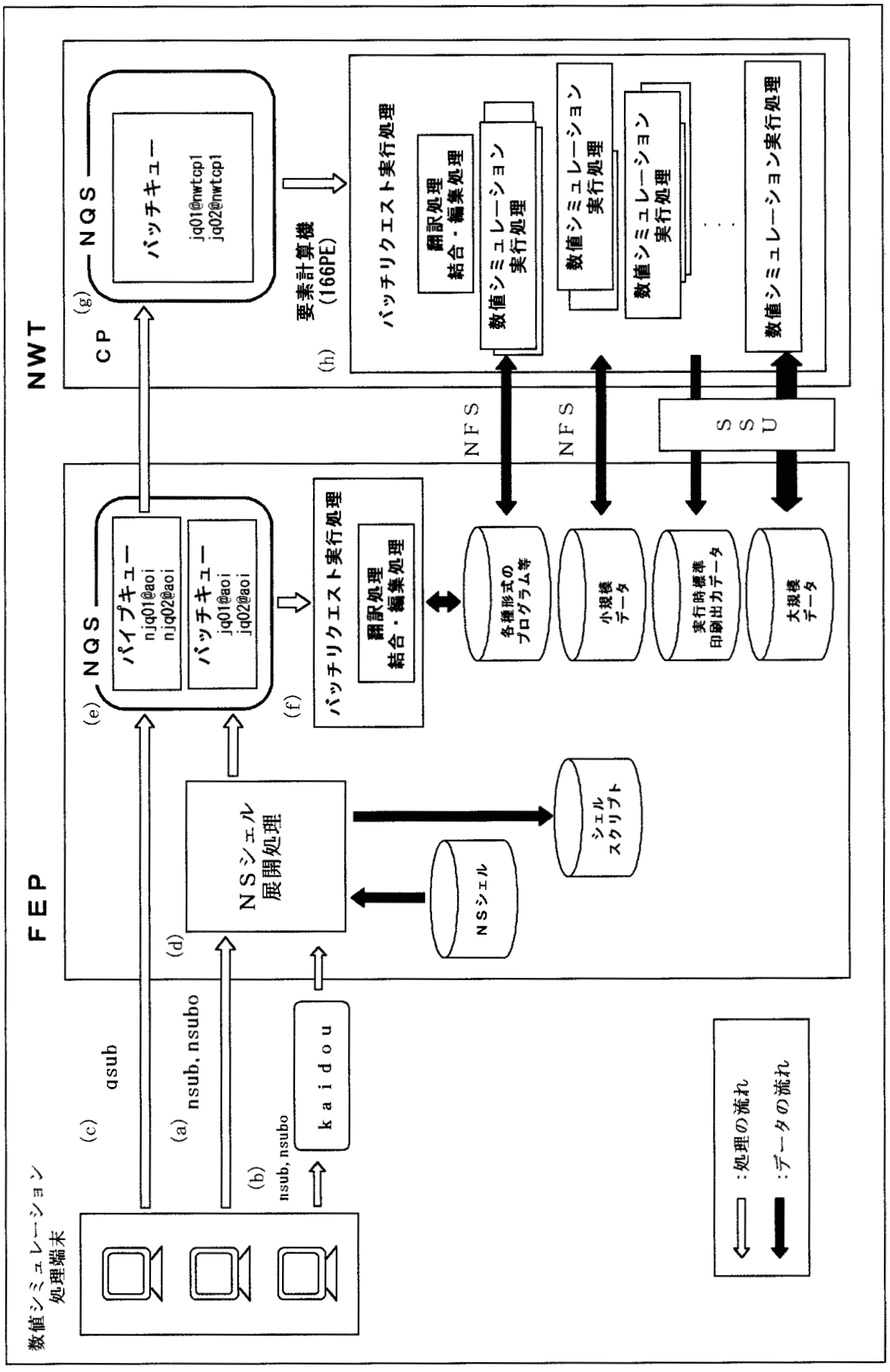

図 5.1 NWT ジョブ処理の流れの概念図

降、総称名をkaidouと呼称する）のセッションから、NSコマンドのnsubまたはnsubo投入から始まる。なお、UNIX本来のバッチリクエスト投入コマンドqsubも利用可能なシステムとしている。NWTジョブの投入方式は以下のとおりである。

(a)は、FEPのセッションからNWTジョブをNSコマンドで投入する流れを示している。予めNWTジョブをNSシェルで記述するファイルを作成・編集してから、nsubまたはnsuboコマンドを投入する。NWTジョブはFEPで受け付けられると、(d)のNSシェルの展開処理が実行される。

(b)は、kaidouのセッションから投入されるNWTジョブの流れを示している。kaidouのセッションからもFEPのセッションと同様に、NSコマンドのnsubおよびnsuboの投入が行える。また、kaidouのセッションでは、nsubおよびnsuboを受け付けると、これらをFEPに転送し、NWTジョブをあらためてFEPに投入する。

(c)は、数値シミュレーション処理端末におけるFEPのセッションからUNIX本来のqsubコマンドでFEPのNQSにバッチリクエストを投入する流れを示している。この場合、NWTジョブを構成する各バッチリクエストを記述する詳細なシェルスクリプトは、予めユーザ自身で作成しておかなければならない。すなわち、バッチリクエストを処理依頼するためのキュー名、NWTジョブ名ならびにバッチリクエスト名等を定義しておかなければならない。さらに、各バッチリクエストの処理を詳細に定義するシェルスクリプトファイルを用意しなければならない。これらの定義は、ジョブ命名規約やシステム資源制限値等、バッチリクエスト走行のための各種運用規約に基づいた正確かつ詳細なシェルスクリプトとする必要がある。

5.2 NSシェルの展開処理

FEPでnsubやnsuboコマンドを受け付けると、図5.1の(d)に示すNSシェルの展開処理が実行される。

NSシェル展開処理では、NWTジョブ投入コマンドで指定されたファイルからNSシェルを読み込み、UNIX本来のバッチリクエストを記述するシェルスクリプトに展開し、当該NWTジョブの特定ファイルにシェルスクリプトを出力する。同時に、展開したシェルスクリプトを実行する。以下にNSシェルの展開処理における主な処理内容を示す。

（1）第4章の第4.1.1項に示したNWTジョブの命名規約に基づいたジョブ名と各バッチリクエストのリクエスト名を定義する。

（2）NSシェルのnjob文に記述されたジョブクラスよりバッチリクエストをキューイングさせるキュー名を以下のとおり決定する。

第4章の第4.1.4項に示すとおり、NSシェルで指定されたジョブクラスがUsual PEクラスのときは、nsub投入時はFEPのバッチキューjq01に、nsubo投入時はパイプキューnjq01を定義する。また、Large PEクラスが指定されているときは、nsub投入時はFEPのバッチキューjq02に、nsubo投入時はパイプキューnjq02を定義し、qsubコマンドに展開する。

（3）NSシェルのnfortc文とnlied文が含まれている場合には、FORTRAN翻訳処理と結合・編集処理のそれぞれプロセスを発行し、1つのバッチリクエストとしてqsubコマンドを定義する。なお、先行するFORTRAN翻訳処理が正常終了を意味する完了コード“0”で復帰しない場合にはエラーとして、以降に処理するバッチリクエストをすべてリジェクトする手続きを挿入している。

（4）NSシェルから展開した各バッチリクエストごとのシェルスクリプトを第4章の表4.3で示した当該NWTジョブ用の一時ワークファイルに出力する。

（5）展開したシェルスクリプトが順次実行し、qsubコマンドが実行されると、NQSにNWTジョブを構成する各バッチリクエストを実行依頼する。

5.3 NQSの処理概要

NQSはqsubコマンドで実行依頼されたバッチリクエストを“待ち行列”で管理し、バッチリクエストを実行起動する。バッチリクエストが終了すると、出力された標準印刷出力ファイルを通常は実行依頼したシステムに転送するが、NWTジョブでは、NSシェルの中で標準印刷出力ファイル単位に第5.5節に示す転送方式でファイルを転送している。以下に(e)のFEPおよび(g)のCPにおけるNQSの処理概要を示す。

（1）バッチキューの資源制限チェックとキューイング

qsubから実行依頼されたバッチリクエストについて、指定されたバッチキュー、jq01またはjq02にキューイングする。このとき、NWTジョブの運用規約に基づいてそれぞれのキューに設定したCPU使用時間やメモリ要求量等のバッチリクエスト資源使用制限値を確認する。

バッチリクエストがこの制限値の条件を満足した場合には、リクエストの情報をNQSが管理するデータベースに登録し、バッチリクエストを指定されたバッチキューに到着順にキューイングする。

バッチリクエストの資源要求量が制限値より超過している場合には、エラー・リクエストとして、これをリジェクトする。同時に、NWTジョブの所有者のユーザセッションにこの旨のメールを送信し、リクエスト情報を削除する。

（2）パイプキューによるバッチリクエスト転送処理

パイプキューnjq01またはnjq02はqsubから実行依頼

されたバッチリクエストを受け取り、これをCPに転送するためのプロセスを発行する。このプロセスの走行制限数はnjq01およびnjq02とも各1に定義している。パイプキューへの到着が多い場合には、バッチリクエストはそれぞれのパイプキューにキューイングされ、到着順に順次、CPへ転送処理される。

(3) バッチリクエストの実行起動処理

qsubによるバッチリクエスト実行依頼があったとき、またはシステムで実行中のバッチリクエストが終了したとき等のジョブスケジューリング契機にNQSはシステムのバッチリクエスト実行総数を確認し、これが制限値に満たない場合には、バッチリクエストを実行起動する。その際、航技研独自のスケジューラ[11]が呼び出され、バッチリクエスト選択のスケジューリング処理が行われ、実行起動すべきバッチリクエストを決定する。NQSはスケジューラが決定したバッチリクエストの実行起動を行う。

5.4 FEPのバッチリクエスト実行処理

nsubで投入されたNWTジョブを構成するバッチリクエストのうち、(f)のFEPのバッチリクエスト処理はクロスコンパイラによるFORTRANプログラム翻訳処理とリンカーの結合・編集処理をFEPで実行する処理フェーズである。なお、FEPでは、NWTのフロントエンドプロセッサとして、良好な応答性を保持し、その役割を十二分に遂行するために、大きな負荷が掛かるユーザプログラム実行処理のバッチリクエストについては定義しない運用システムとしている。

第3.1節に示したとおり、通常の翻訳処理では、ソースプログラムファイルを入力し、実行結果として構文チェック、エラーメッセージならびにソースプログラムリスト等の標準印刷出力ファイルとオブジェクトモジュールを出力するように定義している。また、結合・編集処理では、先行の翻訳処理のプロセスで出力されたオブジェクトモジュールやユーザライブラリおよびシステムライブラリ等を入力し、実行結果として標準印刷出力ファイルとロードモジュール等を出力するように定義する。また、FEPの翻訳処理および結合・編集処理はそれぞれのプロセスを連続して1個のバッチリクエストとして定義して実行処理する。

なお、NSシェル展開処理において作成されたシェルスクリプト中には、もし、FEPのバッチリクエストで重大なエラーが発生した場合、ロードモジュールを作成せず、以降に定義されたバッチリクエスト処理、すなわち数値シミュレーション実行処理をリジェクトする手続きを付加している。

5.5 NWTのバッチリクエスト実行処理

NWTジョブを構成するバッチリクエストのうち、(h)のバッチリクエスト実行処理は、NWTのPEで実行するバッチリクエストである。以下にPEで実行するオウンコンパイラのバッチリクエスト処理とNWTジョブ処理の最終処理フェーズである数値シミュレーション実行処理について述べる。

(1) 翻訳処理と結合・編集処理

nsuboで投入されたNWTジョブは翻訳処理から開始する場合でも、バッチリクエストはFEPのパイプキューからNWTのバッチキューに転送され、すべてのバッチリクエストをNWTで実行する。NWTで実行する翻訳処理と結合・編集処理では1台のPEが割当てられ、それぞれのプロセスは連続して1個のバッチリクエストとして実行処理するように定義している。この処理方式はFEPのクロスコンパイラ処理と同様である。翻訳処理や結合編集処理で入出力するソースプログラムやロードモジュール等の各種形式のユーザプログラムならびに一時ワークファイルはFEP配下の磁気ディスク装置にアクセスし、図5.2のNWT実行時の入出力データの流れに示されるとおり、NFS(注3)でデータ転送する。

(2) 数値シミュレーション実行処理

数値シミュレーション処理のバッチリクエストが実行起動されると、指定されたジョブクラスに対応するPE台数が割当てられ、小規模および大規模の各種入出力ファイルをアクセスしながら計算処理を行う。以下に数値シミュレーション実行処理における入出力データの流れの概要を示す。

NWTは入出力するファイルを直接的に格納する記憶装置を持たないシステムである。バッチリクエスト実行時に入出力するデータの保存場所はFEPの磁気ディスク装置上のファイルである。図5.2に示されるufsファイルシステムはUNIXの標準形式のファイルである。また、vflファイルシステムは大規模データの高速入出力を実現するファイル形式であり、磁気ディスクの入出力性能を上げるためにストライピング機能を使用し、1データを32分割して複数ボリュームに同時転送する。

図5.2に示されるNWT実行時のNFSによるデータ転送では、データはネットワークを経由してFEPとCP間を移動する。このため、その入出力の性能はネットワークの通信負荷に大きく依存するので、数値シミュレーション実行時におけるufsファイルの転送については、小規模データに限って利用する運用とすべきである。

(注3) NFS(Network File System)
ネットワーク接続されたUNIXシステム間においてよく使用されるファイル転送プロトコルである。

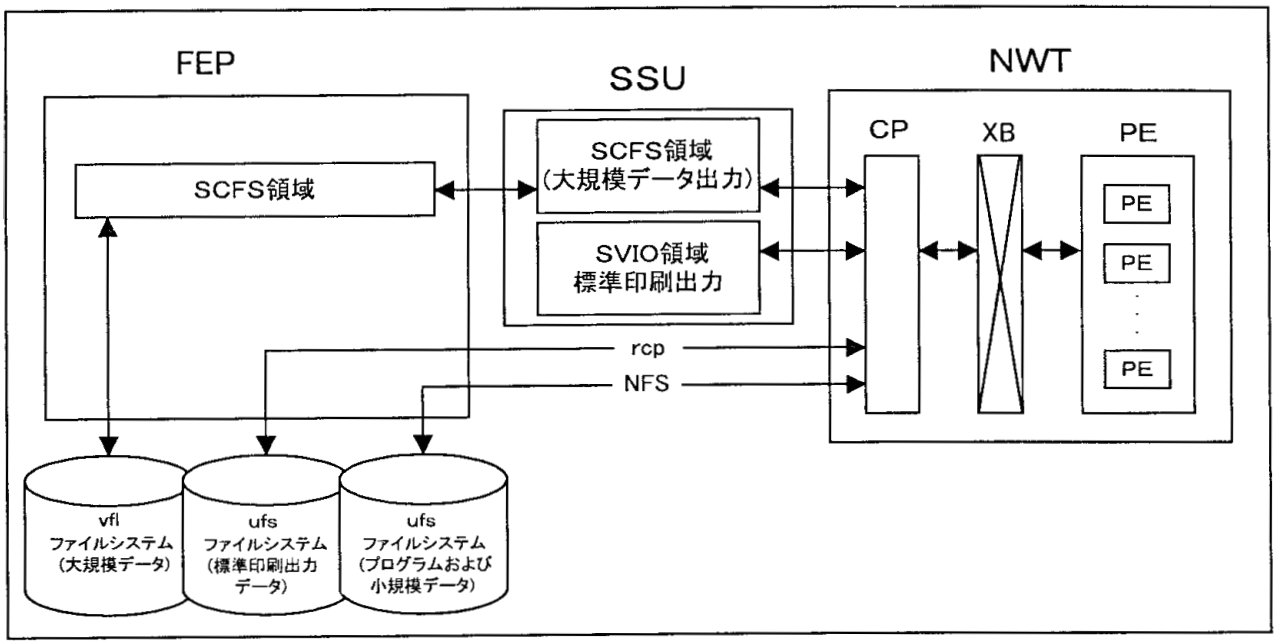

図5.2 NWT実行時の入出力データの流れ

SCFS（Ssu Cashe File System）はSSUをFEPファイルのキャッシュとして使用し、NWTジョブからの大量の入出力を高速に処理することを目的に作られた機能である。さらに、NWTではジョブ実行時に大規模な入力データの読み込み時間を短縮するために、ジョブが実行起動される前に予め入力データをSSUに格納するプレステージング機能を使用している。プレステージングされた入力データがジョブ実行時にもSSUに存在すれば（SSUキャッシュがヒットするという）、改めてFEPのファイルからSSUへデータ転送するための入出力動作は発生しないので、入力時間が非常に短縮できる。実行時の出力データはいったんSSUのSCFS領域に高速に出力され、SSUの1ブロック（8MB単位）がいっぱいになり次第、バッチリクエストの出力処理とは非同期にFEPのメモリのSCFS領域を経由してvflファイルシステムに出力される。NWTジョブでは、SCFSを利用した大規模入出力ファイルの転送処理がNWTジョブの実行における入出力処理の中心となる。

数値シミュレーション処理実行時の標準印刷出力データは直接には磁気ディスクファイルに出力しないで、いったん仮想的な出力イメージでSSUのSVIO領域に高速に出力する。実行終了時にSVIO領域の標準印刷出力データをFEPに書き戻すプログラムをCP上で常に処理可能なデーモンとして定義した。この書き戻しプログラムデーモンはrcpコマンド(注4)を実行し、FEPのufsファイルシステムに標準印刷出力データを書き戻す。なお、標準印刷出力としてSSUのSVIO領域に出力する方式はNFSに比べて入出力性能が非常に高いので、数値シミュレーション実行処理時間の短縮を図ることができ、システムの有効利用性が図れる。

6. NWTジョブ処理の検証と考察

NWTの高度有効利用という目標の基にNWTジョブ処理におけるシステム構築の中から実現した諸機能の概要は第4章に示すとおりである。これらの諸機能はNSシェルとNSコマンドというユーザインターフェースに集約されている。本章では、NSシェルとNSコマンドを介して実現したNWTジョブ処理が所期の目標どおりに所有の機能を確実に遂行しているかどうかを検証するとともに、その有効性について考察する。

6.1 NWTジョブの投入

NSシェルで投入される一般的なNWTジョブのジョブストリーム記述例を図6.1に示す。同図に基づいて以下にNWTジョブの投入方式を検証する。先ず、ここで例示するNWTジョブの処理要求を列挙すると次のとおりである。

(1) FORTRANプログラムの翻訳処理
(2) 結合編集処理
(3) 数値シミュレーション実行処理（その1）
(4) 数値シミュレーション実行処理（その2）
(5) 数値シミュレーション実行処理（その3）

図6.1のnsシェルで指示されるNWTジョブ処理の内

(注4) rcp（remote file copy）
ネットワーク上のマシン間でファイルを複写するコマンド。

```
      njob     u008   ·····①
      nfortc   -p '-Wx -Psd -I/small/x/x01/inc' /small/x/x01/src/*.f   ···②
      nlied    -p '-Wx'   ·····③
   ┌ ngo       -t 18000   ·····④
(1)│ nusdkr    -n 10  /large/x/x01/data10   ·····⑤
   └ nusdkw    -n 20  /large/x/x01/data20   ·····⑥
   ┌ ngo       -t 18000   ·····⑦
(2)│ nusdkr    -n 10  /large/x/x01/data10   ·····⑧
   └ nusdkw    -n 20  /large/x/x01/data20   ·····⑨
   ┌ ngo       -t 18000   ·····⑩
(3)│ nusdkr    -n 10  /large/x/x01/data10   ·····⑪
   └ nusdkw    -n 20  /large/x/x01/data20   ·····⑫
```

図6.1　NSシェルの記述例

容を以下に示す。

①のnjob文は、NWTジョブの先頭を意味するnsシェルである。要素計算機を8台使用する並列ジョブのジョブクラス（u008）を利用することを意味している。

②のnfortc文は、FORTRANコンパイラにより翻訳処理を指示するNSシェルである。－pはこれに続くオプションをFORTRANコンパイラに渡すことを指示するオプションである。コンパイラに渡すオプションの内、－Wxは並列プログラム処理、－Psdはソースプログラムリストの出力を指示している。また、－Iはインクルードオプション(注5)である。これに続く/small/x/x01/incはインクルードファイル名を意味している。/small/x/x01/src/*.fは翻訳処理するソースプログラムが格納されているファイル名を意味する。

③のnlied文は、結合・編集処理を指示する制御文である。－pはこれに続くオプションをリンカーに渡すことを指示するオプションである。リンカーに渡される－Wxは並列プログラム処理を意味する。

④のngo文は、NWTにおける数値シミュレーション実行処理を指示するnsシェルである。－tはCPU打ち切り時間（単位：秒）を指定するオプションで、続けて指定時間を数字で記述する。この例では18000秒が指定されている。

⑤のnusdkr文は、④で指示する実行処理の中で参照するファイルを定義するNSシェルである。－nはreadする論理ファイル機番の番号を意味し、これに続けて2桁の番号（この例では10）とファイル名（この例では、/large/x/x01/data10）を指定する。

⑥のnusdkw文は、④で指示する実行処理の中で出力するファイルを定義するNSシェルである。－nはwriteする論理ファイル機番の指示を意味し、これに続けて2桁の番号（この例では20）とファイル名（この例では、/large/x/x01/data20）を指定する。

⑦と⑩は④と等価であり、数値シミュレーション実行処理を指示するnsシェルである。同じnjob文の中で複数定義される数値シミュレーション実行処理は、定義される順序で順番に実行することを意味する。すなわち、④～⑥で定義されている数値シミュレーション実行処理（その1）が終了すると、⑦～⑨の実行処理（その2）を実行可能にする。また、（その2）の数値シミュレーション実行処理が終了すると、⑩～⑫で定義されている実行処理（その3）を実行可能にする。NWTでは、このように数値シミュレーション実行処理を多数連続して実行しなければ解を得られないような長時間ジョブが多数発生する。なお、このように連続処理するバッチリクエストの実行起動は航技研独自のスケジューラに制御を渡しており、バッチリクエストを順序正しくかつ順番に実行起動する仕組みとしている。

⑧と⑪は⑤と等価である。

⑨と⑫は⑥と等価である。

上記から、図6.1で示すNSシェルは今まで慣れ親しんできた従来システムのMSPのNSカタプロ[3]記述形式と非常に酷似している。すなわち、①から⑫に示すように、各NSシェルはNSカタプロと1対1に対応し、NSカタプロ名と同じシェル名を持つものを用意している。また、NSシェルの記述する順序もNSカタプロのそれと同様に行える。

また、UNIX本来のシェルスクリプトで図6.1のジョブを記述すると、図6.2のとおりである。二つの図で比較す

（注5）インクルードオプション
FORTRANプログラムの中で一部の手続きを別ファイルに作成し、このファイルをプログラム中に挿入することを指示するオプション

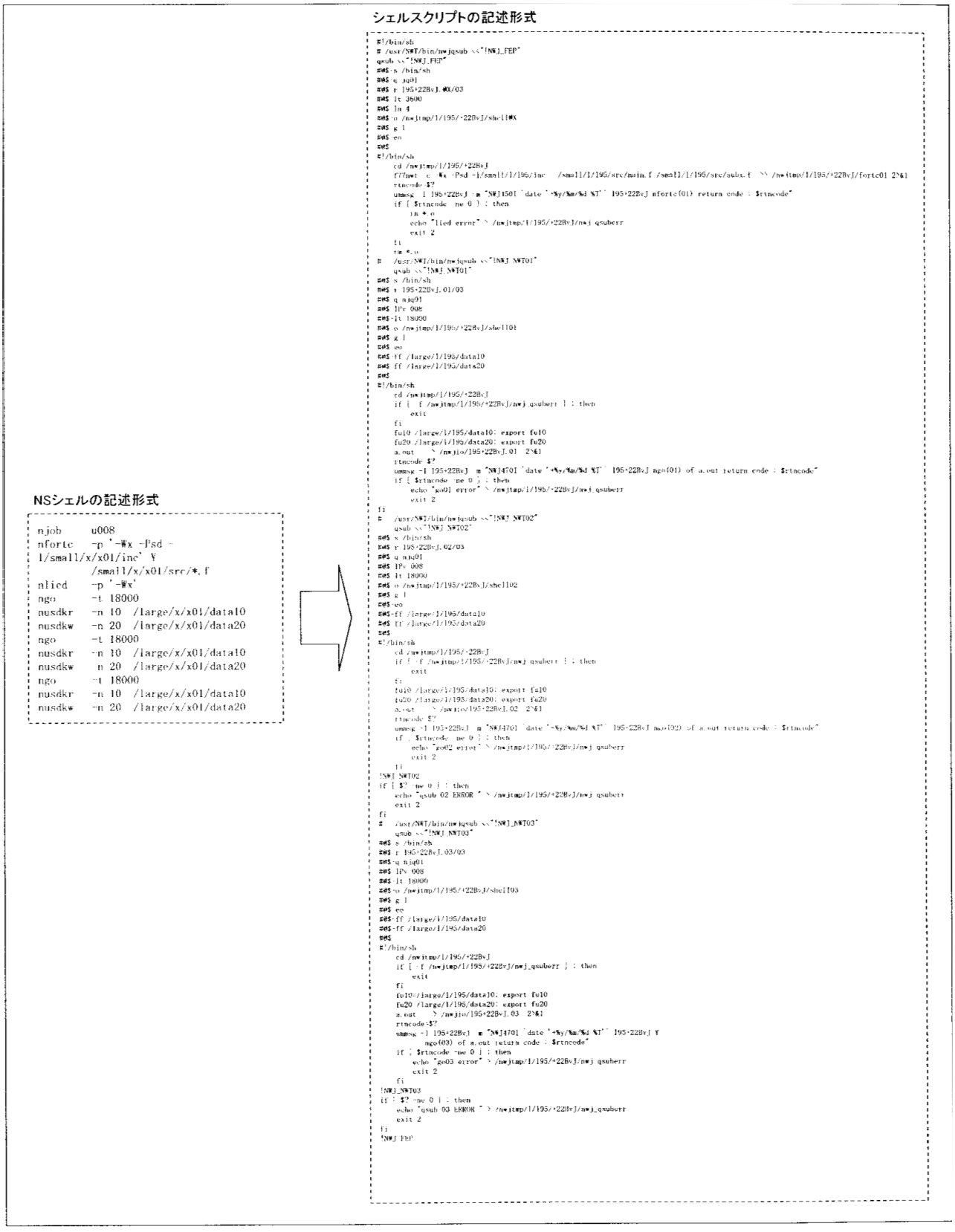

図 6.2 NS シェルからシェルスクリプトへの展開例

```
aoi% nsub ns-shell
Request 35612.aoi submitted to queue: jq01.    ·····①
UX:nsubx: 情報: job name is x01+10DIY.    ·····②
aoi%
```

図6.3 NWTジョブの投入確認

ると一目瞭然であるように、NSシェルは必要となる各種の定義を極端に省略し得るので、NWTジョブの記述は非常に容易かつ確実に行えると判断でき、システム利用性向上に有効となると考えられる。

6.2 NWTジョブの投入確認

NSシェルで記述したNWTジョブをNSコマンドのnsubでシステムに投入したとき、図6.3に示すプロンプト(注6)がユーザ端末にエコーされる。すなわち、同図は図6.1の内容のNSシェルをns-shellというファイルに保存し、nsubコマンドを実行したときの端末表示結果である。図6.3の①からNWTジョブは正常にサブミットされ、FEPのバッチキューjq01（jq01@aoiの意味）にキューイングされたことが理解できる。なお、35612.aoiはUNIXシステムの標準機能が付加するバッチリクエスト追番である。また、同図の②から、ユーザ名および投入日時等より、x01＋10DIYというNWTジョブ名が発行されたことが分かる。以上から、NSシェル展開処理においてジョブ命名規約に従ってNWTジョブのジョブ名が設定されていることが確認できる。nsubコマンド投入された複数バッチリクエストで構成される検証ジョブは、上記ジョブ名を有するNWTジョブという単位で管理され、実行処理が経過していくことになる。また、ユーザはこのジョブ名を基にNWTジョブの実行処理状況や各種の印刷イメージの標準印刷出力ファイルの検索および取り出しが行えることになる。

6.3 NWTジョブの実行経過状況の確認

NWTジョブは処理の経過とともに、その実行状況を示す各種のログ情報を時々刻々とファイルに出力する。このファイルは第4.1.5項で示したとおり一時ワークファイルとして作成される。検証ジョブのログ情報の内容を図6.4に示す。また、図6.4の各メッセージの意味を図6.5に示す。このログ情報はユーザのセッションからNSコマンドのnlogを投入すると得られる。nlogコマンドについては第6.6節で説明する。これらの図に示すとおり、NWTジョブの実行処理状況がジョブの所有者であるユーザのセッションで確認でき、ユーザがNSシェルで指示するとおり、NWTジョブを構成するバッチリクエスト列が正確な実行順序で順番に実行されていく様子が確認できる。換言すると、図6.1に示した検証ジョブの実行処理経過から、ユーザが記述するNSシェルに集約された諸機能が所期の目標のとおり所有の機能を遂行していることを示すものであると判断できる。

6.4 NWTジョブの実行待ち状況の確認

NWTにおける数値シミュレーション実行処理を起動待ちしている実行可能ジョブの状況はwaitjobコマンドの結果として得られる。図6.6はwaitjobの実行結果を示す。同図から実行待ちしているジョブについて、属するキュー名、NWTジョブの優先順序、NWTジョブ名、バッチリクエスト名、リクエストID、要求PE台数、要求CPU使用時間、NWTジョブ受付時間、実行待ちの各種状態等が確認できる。ここでキュー名u0はNWTのバッチキューjq01＠nwtcp1を示す。また、同様にl0はjq02＠nwtcp1を意味する。この表示の中で、検証ジョブは図6.3に示したNWTジョブ名(X01+10DIY)、バッチリクエスト名(01/03～03/03）を有し、ジョブの優先順序は21番として表示されている。また、同ジョブは図6.1のNSシェルに示したとおり3つの数値シミュレーション実行処理がキューイングされている状況が確認できる。時刻が経過し、検証ジョブが実行起動されるスケジューリングの契機が来ると、先頭のバッチリクエストから順次起動されていく。なお、下位の5行の表示の上3行は、連続する実行待ちジョブの表示に続けて次回ジョブスケジューリング契機に実行起動を予定しているジョブについて、要求PE台数と実行開始予定時刻等を表示している。以上のとおり、NSコマンドのwaitjobはNWTジョブの実行待ち状況を分かり易く表示し、ジョブのターン・アラウンド・タイムの予測を可能にし、また、システム内のジョブ混雑状況を明確に表示していると判断できる。

(注6) プロンプト
会話処理においてシステムが応答するメッセージであり、プロンプト送出の後はユーザの入力モードとなる。

```
*****************************************************************
* nsubx ns-shell
* submit on /small/x/x01
*****************************************************************

njob     u008
nfortc   -p '-Wx -Psd -I/small/x/x01/inc' /small/x/x01/src/*.f
nlied    -p '-Wx'
ngo      -t 18000
nusdkr   -n 10  /large/x/x01/data10
nusdkw   -n 20  /large/x/x01/data20
ngo      -t 18000
nusdkr   -n 10  /large/x/x01/data10
nusdkw   -n 20  /large/x/x01/data20
ngo      -t 18000
nusdkr   -n 10  /large/x/x01/data10
nusdkw   -n 20  /large/x/x01/data20

*****************************************************************

NWX101I 00/02/10 13:18:39 x01+10DIY.@X/03,35612.aoi,jq01,M,ACCEPT:UXP
NWX201I 00/02/10 13:18:39 x01+10DIY.@X/03,35612.aoi,jq01,START:UXP
NWJ450I 00/02/10 13:18:42 x01+10DIY nfortc(01) return code : 0:UXP
NWJ460I 00/02/10 13:18:45 x01+10DIY nlied(ld) return code : 0:UXP
NWX101I 00/02/10 13:18:45 x01+10DIY.01/03,35613.aoi,jq01,H,ACCEPT:NWT
NWX301I 00/02/10 13:18:47 x01+10DIY.@X/03,35612.aoi,END,246/100:UXP
NWX910I REQUEST NAME    QUE    ACCEPT   TIME     START    TIME      E N D ····· TIME:UXP
NWX911I x01+10DIY.@X/03 jq01 00/ 2/10 13:18:39 00/ 2/10 13:18:39 00/ 2/10 ····· 2.3344:UXP
NWX101I 00/02/10 13:18:49 x01+10DIY.02/03,35614.aoi,jq01,H,ACCEPT:NWT
NWX101I 00/02/10 13:18:52 x01+10DIY.03/03,35615.aoi,jq01,H,ACCEPT:NWT
NWX201I 00/02/10 13:28:51 x01+10DIY.01/03,35613.aoi,jq01,START:NWT
NWJ470I 00/02/10 13:34:03 x01+10DIY ngo(01) of a.out return code : 0:NWT
NWX201I 00/02/10 13:34:04 x01+10DIY.02/03,35614.aoi,jq01,START:NWT
NWX301I 00/02/10 13:34:06 x01+10DIY.01/03,35613.aoi,END,239920/100:NWT
NWJ500I 00/02/10 13:34:06 go01 : write back complete:UXP
NWX910I REQUEST NAME    QUE    ACCEPT   TIME     START    TIME      E N D ····· TIME:NWT
NWX911I x01+10DIY.01/03 u008 00/ 2/10 13:18:48 00/ 2/10 13:28:51 00/ 2/10 ···· 2399.4056:NWT
NWX912I VU TIME USER        SYSTEM:NWT
NWX914I       87.6642       0.0000:NWT
NWX912I PE NO. PARA/MAST   ELAPSE     MEMORY(V)    MEMORY(R):NWT
NWX915I  0000  PARA/MAST    310.8631      91328       83136 :NWT
NWX915I  0001  PARA         300.2388      25792       17600 :NWT
NWX915I  0007  PARA         300.0383      25792       17600 :NWT
NWX915I  0004  PARA         300.0395      25792       17600 :NWT
NWX915I  0002  PARA         300.1543      25792       17600 :NWT
NWX915I  0006  PARA         300.0457      25792       17600 :NWT
NWX915I  0003  PARA         300.0444      25792       17600 :NWT
NWX915I  0005  PARA         300.1603      25792       17600 :NWT
NWX915I   CP   NOT PARA     312.6313       1888             :NWT
NWJ470I 00/02/10 13:39:10 x01+10DIY ngo(02) of a.out return code : 0:NWT
NWX201I 00/02/10 13:39:11 x01+10DIY.03/03,35615.aoi,jq01,START:NWT
NWX301I 00/02/10 13:39:13 x01+10DIY.02/03,35614.aoi,END,239931/100:NWT
NWX910I REQUEST NAME    QUE    ACCEPT   TIME     START    TIME      E N D ····· TIME:NWT
NWX911I x01+10DIY.02/03 u008 00/ 2/10 13:18:51 00/ 2/10 13:34:04 00/ 2/10 ···· 2399.5094:NWT
NWX912I VU TIME USER        SYSTEM:NWT
NWX914I       87.6645       0.0000:NWT
NWX912I PE NO. PARA/MAST   ELAPSE     MEMORY(V)    MEMORY(R):NWT
NWX915I  0000  PARA/MAST    305.3463      91328       83136 :NWT
NWX915I  0001  PARA         300.2791      25792       17600 :NWT
NWX915I  0007  PARA         300.0753      25792       17600 :NWT
NWX915I  0004  PARA         300.0763      25792       17600 :NWT
NWX915I  0002  PARA         300.1926      25792       17600 :NWT
NWX915I  0006  PARA         300.0825      25792       17600 :NWT
NWX915I  0003  PARA         300.0816      25792       17600 :NWT
NWX915I  0005  PARA         300.1978      25792       17600 :NWT
NWX915I   CP   NOT PARA     306.9848       1888             :NWT
NWJ470I 00/02/10 13:44:18 x01+10DIY ngo(03) of a.out return code : 0:NWT
NWX301I 00/02/10 13:44:20 x01+10DIY.03/03,35615.aoi,END,239928/100:NWT
NWX910I REQUEST NAME    QUE    ACCEPT   TIME     START    TIME      E N D ····· TIME:NWT
NWX911I x01+10DIY.03/03 u008 00/ 2/10 13:18:55 00/ 2/10 13:39:11 00/ 2/10 ···· 2399.4772:NWT
NWX912I VU TIME USER        SYSTEM:NWT
NWX914I       87.6646       0.0000:NWT
NWX912I PE NO. PARA/MAST   ELAPSE     MEMORY(V)    MEMORY(R):NWT
NWX915I  0000  PARA/MAST    306.2798      91328       83136 :NWT
NWX915I  0001  PARA         300.2782      25792       17600 :NWT
NWX915I  0007  PARA         300.0708      25792       17600 :NWT
NWX915I  0002  PARA         300.1913      25792       17600 :NWT
NWX915I  0004  PARA         300.0773      25792       17600 :NWT
NWX915I  0003  PARA         300.0757      25792       17600 :NWT
NWX915I  0006  PARA         300.0775      25792       17600 :NWT
NWX915I  0005  PARA         300.1972      25792       17600 :NWT
NWX915I   CP   NOT PARA     308.3004       1888             :NWT
```

図 6.4　NWT ジョブのログ情報

```
①  ****************************************************************
    * nsubx ns-shell
    * submit on /small/x/x01
    ****************************************************************
      ① NSシェル投入であることを示すメッセージ
②   njob     u008
    nfortc   -p '-Wx -Psd -I/small/x/x01/inc' /small/x/x01/src/*.f
    nlied    -p '-Wx'
    ngo      -t 18000
    nusdkr   -n 10  /large/x/x01/data10
    nusdkw   -n 20  /large/x/x01/data20
    ngo      -t 18000
    nusdkr   -n 10  /large/x/x01/data10
    nusdkw   -n 20  /large/x/x01/data20
    ngo      -t 18000
    nusdkr   -n 10  /large/x/x01/data10
    nusdkw   -n 20  /large/x/x01/data20

    ****************************************************************
      ② NSシェルの内容を示す。
③   NWX101I 00/02/10 13:18:39 x01+10DIY.@X/03,35612.aoi,jq01,M,ACCEPT:UXP
    NWX201I 00/02/10 13:18:39 x01+10DIY.@X/03,35612.aoi,jq01,START:UXP
    NWJ450I 00/02/10 13:18:42 x01+10DIY nfortc(01) return code : 0:UXP
    NWJ460I 00/02/10 13:18:45 x01+10DIY nlied(ld) return code : 0:UXP
      ③ 翻訳、結合・編集バッチリクエストの受付、実行開始、完了コードを示す。
④   NWX101I 00/02/10 13:18:45 x01+10DIY.01/03,35613.aoi,jq01,H,ACCEPT:NWT
      ④ 数値シミュレーション実行処理（その１）のバッチキュー受付を示す。
⑤   NWX301I 00/02/10 13:18:47 x01+10DIY.@X/03,35612.aoi,END,246/100:UXP
    NWX910I REQUEST NAME    QUE    ACCEPT   TIME     START    TIME     E N D ･･･････ TIME:UXP
    NWX911I x01+10DIY.@X/03 jq01 00/ 2/10 13:18:39 00/ 2/10 13:18:39 00/ 2/10 ･･･････ 2.3344:UXP
      ⑤ 翻訳、結合・編集バッチリクエスト終了、処理時刻およびCPU使用時間を示す。
⑥   NWX101I 00/02/10 13:18:49 x01+10DIY.02/03,35614.aoi,jq01,H,ACCEPT:NWT
      ⑥ 数値シミュレーション実行処理（その２）のバッチキュー受付を示す。
⑦   NWX101I 00/02/10 13:18:52 x01+10DIY.03/03,35615.aoi,jq01,H,ACCEPT:NWT
      ⑦ 数値シミュレーション実行処理（その３）のバッチキュー受付を示す。
⑧   NWX201I 00/02/10 13:28:51 x01+10DIY.01/03,35613.aoi,jq01,START:NWT
      ⑧ 数値シミュレーション実行処理（その１）の実行開始を示す。
⑨   NWJ470I 00/02/10 13:34:03 x01+10DIY ngo(01) of a.out return code : 0:NWT
      ⑨ 数値シミュレーション実行処理（その１）の完了コードを示す。
⑩   NWX201I 00/02/10 13:34:04 x01+10DIY.02/03,35614.aoi,jq01,START:NWT
      ⑩ 数値シミュレーション実行処理（その２）の実行開始を示す。
⑪   NWX301I 00/02/10 13:34:06 x01+10DIY.01/03,35613.aoi,END,239920/100:NWT
      ⑪ 数値シミュレーション実行処理（その１）の実行終了、CPU使用時間（単位100分の1秒）を示す。
⑫   NWJ500I 00/02/10 13:34:06 go01 : write back complete:UXP
      ⑫ 数値シミュレーション実行処理（その１）の標準印刷出力の書き戻し完了を示す。
⑬   NWX910I REQUEST NAME    QUE    ACCEPT   TIME     START    TIME     E N D ･･･････ TIME:NWT
    NWX911I x01+10DIY.01/03 u008 00/ 2/10 13:18:48 00/ 2/10 13:28:51 00/ 2/10 ････ 2399.4056:NWT
    NWX912I VU TIME USER         SYSTEM:NWT
    NWX914I       87.6642        0.0000:NWT
    NWX912I PE NO. PARA/MAST   ELAPSE     MEMORY(V)     MEMORY(R):NWT
    NWX915I  0000  PARA/MAST    310.8631      91328       83136 :NWT
    NWX915I  0001  PARA         300.2388      25792       17600 :NWT
    NWX915I  0007  PARA         300.0383      25792       17600 :NWT
    NWX915I  0004  PARA         300.0395      25792       17600 :NWT
    NWX915I  0002  PARA         300.1543      25792       17600 :NWT
    NWX915I  0006  PARA         300.0457      25792       17600 :NWT
    NWX915I  0003  PARA         300.0444      25792       17600 :NWT
    NWX915I  0005  PARA         300.1603      25792       17600 :NWT
    NWX915I   CP   NOT PARA     312.6313       1888             :NWT
      ⑬ 数値シミュレーション実行処理（その１）の実行終了、処理時刻およびCPU使用時間ならびに
        各PEの資源使用状況を示す。
⑭   NWJ470I 00/02/10 13:39:10 x01+10DIY ngo(02) of a.out return code : 0:NWT
      ⑭ 数値シミュレーション実行処理（その２）の完了コードを示す。
⑮   NWX201I 00/02/10 13:39:11 x01+10DIY.03/03,35615.aoi,jq01,START:NWT
      ⑮ 数値シミュレーション実行処理（その３）の実行開始を示す。
⑯   NWX301I 00/02/10 13:39:13 x01+10DIY.02/03,35614.aoi,END,239931/100:NWT
      ⑯ 数値シミュレーション実行処理（その２）の実行終了、CPU使用時間を示す。
```

図6.5　NWTジョブのログ情報の内容

```
NWX910I REQUEST NAME    QUE    ACCEPT   TIME     START    TIME      E N D ····· TIME:NWT
NWX911I x01+10DIY.02/03 u008 00/ 2/10 13:18:51 00/ 2/10 13:34:04 00/ 2/10 ···· 2399.5094:NWT
NWX912I VU TIME USER         SYSTEM:NWT
NWX914I       87.6645        0.0000:NWT
NWX912I PE NO.  PARA/MAST   ELAPSE     MEMORY(V)    MEMORY(R):NWT
NWX915I  0000   PARA/MAST    305.3463      91328       83136 :NWT
NWX915I  0001   PARA         300.2791      25792       17600 :NWT
NWX915I  0007   PARA         300.0753      25792       17600 :NWT
NWX915I  0004   PARA         300.0763      25792       17600 :NWT
NWX915I  0002   PARA         300.1926      25792       17600 :NWT
NWX915I  0006   PARA         300.0825      25792       17600 :NWT
NWX915I  0003   PARA         300.0816      25792       17600 :NWT
NWX915I  0005   PARA         300.1978      25792       17600 :NWT
NWX915I   CP    NOT PARA     306.9848       1888             :NWT
```

⑰ 数値シミュレーション実行処理（その２）の実行終了、処理時刻およびCPU使用時間ならびに各PEの資源使用状況を示す。

```
NWJ470I 00/02/10 13:44:18 x01+10DIY ngo(03) of a.out return code : 0:NWT
```

⑱ 数値シミュレーション実行処理（その３）の完了コードを示す。

```
NWX301I 00/02/10 13:44:20 x01+10DIY.03/03,35615.aoi,END,239928/100:NWT
```

⑲ 数値シミュレーション実行処理（その３）の実行終了、CPU使用時間を示す。

```
NWX910I REQUEST NAME    QUE    ACCEPT   TIME     START    TIME      E N D ····· TIME:NWT
NWX911I x01+10DIY.03/03 u008 00/ 2/10 13:18:55 00/ 2/10 13:39:11 00/ 2/10 ···· 2399.4772:NWT
NWX912I VU TIME USER         SYSTEM:NWT
NWX914I       87.6646        0.0000:NWT
NWX912I PE NO.  PARA/MAST   ELAPSE     MEMORY(V)    MEMORY(R):NWT
NWX915I  0000   PARA/MAST    306.2798      91328       83136 :NWT
NWX915I  0001   PARA         300.2782      25792       17600 :NWT
NWX915I  0007   PARA         300.0708      25792       17600 :NWT
NWX915I  0002   PARA         300.1913      25792       17600 :NWT
NWX915I  0004   PARA         300.0773      25792       17600 :NWT
NWX915I  0003   PARA         300.0757      25792       17600 :NWT
NWX915I  0006   PARA         300.0775      25792       17600 :NWT
NWX915I  0005   PARA         300.1972      25792       17600 :NWT
NWX915I   CP    NOT PARA     308.3004       1888             :NWT
```

⑳ 数値シミュレーション実行処理（その３）の実行終了、処理時刻およびCPU使用時間ならびに各PEの資源使用状況を示す。

図6.5 NWTジョブのログ情報の内容 つづき

```
Q    No.   REQUEST-NAME     REQUEST-ID   PE    CPU ACCEPT-TIME  STATUS
u0     1.  e80+101Hb.05/06 35389.aoi      6    300 10-01:19:03 next-step
           e80+101Hb.06/06 35390.aoi      6    300 10-01:19:07 next-step
       2.  b62-J0789.08/08 34223.aoi     30  18000 08-09:28:57 next-step
       3.  q89+03CHN.06/10 32568.aoi      4  20000 03-12:18:41 next-step
           q89+03CHN.07/10 32569.aoi      4  20000 03-12:18:45 next-step
           q89+03CHN.08/10 32570.aoi      4  20000 03-12:18:50 next-step
           q89+03CHN.09/10 32571.aoi      4  20000 03-12:18:53 next-step
           q89+03CHN.10/10 32572.aoi      4  20000 03-12:18:57 next-step
                  :                      :                    :
      18.  a20+10Ai9.01/01 35564.aoi     11  14000 10-10:48:41 pe-busy
      19.  q67+04FNP.01/04 33107.aoi     64   6000 04-15:23:31 pe-busy
           q67+04FNP.02/04 33108.aoi     64   6000 04-15:23:34 next-step
           q67+04FNP.03/04 33109.aoi     64   6000 04-15:23:38 next-step
           q67+04FNP.04/04 33110.aoi     64   6000 04-15:23:41 next-step
      20.  131+105ou.01/01 35477.aoi     21  16000 10-05:52:10 pe-busy
      21.  x01+10DIY.01/03 35613.aoi      8  18000 10-13:18:45 pe-busy
           x01+10DIY.02/03 35614.aoi      8  18000 10-13:18:49 next-step
           x01+10DIY.03/03 35615.aoi      8  18000 10-13:18:52 next-step
      22.  131+106Eh.01/01 35479.aoi     31  20000 10-06:15:56 upe-limit
                  :                      :                    :
      40.  p40+108hv.01/01 35494.aoi      1   4800 10-08:44:01 resv-time
      41.  p40+108iv.01/01 35496.aoi      1   4800 10-08:45:02 resv-time
      42.  p40+108jL.01/01 35498.aoi      1   4800 10-08:45:26 resv-time
      43.  p42+109Wo.04/04 35508.aoi      1   2500 10-09:33:10 next-step
      44.  p42+10An0.01/02 35565.aoi      1   2500 10-10:49:33 usr-limit
           p42+10An0.02/02 35566.aoi      1   2500 10-10:49:38 next-step
      45.  p42+10AnV.01/02 35567.aoi      1   2500 10-10:49:43 usr-limit
           p42+10AnV.02/02 35568.aoi      1   2500 10-10:49:50 next-step
10     1.  164+08Kud.03/05 34797.aoi      1  20000 08-20:56:52 pe-busy
           164+08Kud.04/05 34798.aoi      1  20000 08-20:56:56 pe-busy
           164+08Kud.05/05 34799.aoi      1  20000 08-20:57:00 pe-busy
       2.  164+09Atw.01/05 34967.aoi      1  20000 09-10:56:04 pe-busy
           164+09Atw.02/05 34968.aoi      1  20000 09-10:56:08 pe-busy
           164+09Atw.03/05 34969.aoi      1  20000 09-10:56:12 pe-busy
           164+09Atw.04/05 34970.aoi      1  20000 09-10:56:16 pe-busy
           164+09Atw.05/05 34971.aoi      1  20000 09-10:56:20 pe-busy
-----
INFO: 10pe reserved for j15+10DDI.01/01.
INFO: The job will be started at 13:33:41 restricted to 2500sec-cpu time.
INFO: But job(pe<=7 & cpu<781sec) is excutable now.
INFO: AOI REQUEST(compile) NOT FOUND.
INFO: AOI REQUEST(routing) NOT FOUND.
```

図 6.6　waitjob コマンドの表示結果

6.5 NWTジョブの実行状況

NWTジョブの実行状況を表示することを指示するNSコマンドのactjobの実行結果を図6.7に示す。同図に示すとおり、NWT実行中ジョブ状況が確認でき、実行ジョブについて、キュー名、NWTジョブ名、バッチリクエスト名およびリクエストID、実行状態、割当PE台数、実使用PE台数、要求CPU使用時間、ジョブ経過時間等を表示する。実使用PE台数の表示において、1PEを使用するジョブについては並列ジョブでないために0で表示されるが実際は1PEを使用している。なお、最下行の表示は、PEの稼働状況について示し、運用台数、ONLINE／OFFLINE台数、実行／非実行台数、および1GBのジョブ用に割り当てているPE台数を表示している。同図の実行状況から、NWTの166台のPEはすべて実行中のバッチリクエストに割り当てられており、100%の稼働率を呈している。

ジョブ名の降順に表示された実行中ジョブを確認すると、上から18番目に検証ジョブの数値シミュレーション実行処理（その1）が確認できる。

以上のとおり、actjobとwaitjobコマンドを確認すると、ユーザは投入ジョブのターン・アラウンド・タイムの予測が可能となり、これらの情報はユーザの数値シミュレーション処理プログラムの開発および処理の計画に有効な機能であると判断できる。

6.6 NWTジョブの実行結果の検索

第4.1.5項で示すとおり、NWTジョブの処理の過程ではログ情報ならびに各バッチリクエストごとの標準印刷出力結果がNWTジョブごとに割り当てられた一時ワークファイルに順次出力される。これらの印刷イメージの出力結果を検索するための機能としてnlogコマンドが用意されている。ユーザのセッションからnlogコマンドを投入すると、その時点までに出力されている上記の各種出力結果を検索することができる。図6.8はnlogコマンドを実行したときの結果を示す。同図に基づいてnlogコマンドの表示結果について説明する。

①nlogを投入すると、その時点におけるユーザのNWTジョブ名の一覧が番号付きで表示される。

②番号4に示されるNWTジョブ（検証ジョブ）の検索を指示するために入力域に4を入力すると、当該ジョブ

```
date : 00/02/10-13:30:34
-----
QUE   REQUEST_NAME   REQUEST_ID     STATUS ALLOC USE    REQUEST_CPU      TIME
jq01 a06s599_-08cd03 2253.tenjin       RUN     1   0  18000( 5:00:00) 2:20:38
jq01 a06s600_-08ce03 2254.tenjin       RUN     1   0  18000( 5:00:00) 2:20:36
jq01 a06s601_-08cf03 2255.tenjin       RUN     1   0  18000( 5:00:00) 1:45:51
jq01 a12+09FQC.03/05 35140.aoi         RUN    16  16  17800( 4:56:40) 0:24:13
jq01 a12+09Fa8.02/05 35146.aoi         RUN    16  16  17800( 4:56:40) 4:00:10
jq01 a31+109gR.05/07 35521.aoi         RUN     1   0   3600( 1:00:00) 0:09:01
jq01 b46+0894w.10/10 34209.aoi         RUN    10  10  20000( 5:33:20) 1:19:42
jq01 b46+09FeU.03/05 35152.aoi         RUN    10  10  20000( 5:33:20) 2:57:57
jq01 b62-J0789.07/08 34222.aoi         RUN    30  30  18000( 5:00:00) 1:02:32
jq01 b77+09LKU.03/09 35350.aoi         RUN    16  16  10000( 2:46:40) 1:19:24
jq01 e80+101Hb.04/06 35388.aoi         RUN     6   6  18000( 5:00:00) 0:44:40
jq01 f45+10ADf.02/10 35550.aoi         RUN     1   0   7200( 2:00:00) 0:07:48
jq01 j06+09CY8.07/20 35065.aoi         RUN    12  12  15000( 4:10:00) 1:44:06
jq01 j62+109Dw.01/01 35503.aoi         RUN     1   0  20000( 5:33:20) 4:09:19
jq01 j62+10AOL.01/01 35560.aoi         RUN     1   0  10500( 2:55:00) 2:20:41
jq01 l31+084t3.18/18 34171.aoi         RUN    19  19  12000( 3:20:00) 0:43:46
jq01 l32+109wk.03/05 35532.aoi         RUN     7   7   3000( 0:50:00) 0:46:18
jq01 x01+10DIY.01/03 35613.aoi         RUN     8   8  18000( 5:00:00) 0:01:09
jq01 p42+109Wo.03/04 35507.aoi         RUN     1   0   2500( 0:41:40) 0:24:14
jq01 p42+109Xa.02/02 35510.aoi         RUN     1   0   2500( 0:41:40) 0:24:12
jq01 p42+109Xk.02/02 35512.aoi         RUN     1   0   2500( 0:41:40) 0:24:07
jq01 p42+109Xu.02/02 35514.aoi         RUN     1   0   2500( 0:41:40) 0:24:06
jq01 q89+03CHN.05/10 32567.aoi         RUN     4   4  20000( 5:33:20) 0:36:54
jq02 l64+08Kud.02/05 34796.aoi         RUN     1   0  20000( 5:33:20) 0:21:41

TOTAL_PE   ONLINE  OFFLINE   RUNNING[OFFLINE]   IDLING[ONLINE]  1G[ONLINE]
   166       166        0        166       0        0       0      1       1
```

図6.7 actjobコマンドの表示結果

の既に出力済みの各種出力情報を番号付きで表示する。

③入力域に検索すべき出力情報の番号2（数値シミュレーション実行処理その1を意味する）を入力する。

④出力結果の表示位置を示す（結果の表示は省略）。

⑤入力域に番号2の出力情報を印刷（Printer）を指示することを示す。

⑥プリンター名の入力を促すプロンプトを返すので、入力域に印刷を取り出したいプリンター名を入力する。

以上のとおり、nlogコマンドの表示結果から、第4.1.5項で示したNWTジョブの各種出力情報を出力する機能が有効であると判断できる。

6.7 考察

検証ジョブにより、NWTジョブをその投入から実行結果の取り出しに至るまでのシステム内ジョブ処理状況を克明に示し、本システム構築により実現したジョブ処理機能を検証した。上記の検証結果に基づいて以下のとおり考察する。

(1) NSシェルやNSコマンドを利用すると、NWTジョブ処理は容易かつ確実なものとなり、UNIXの利用に慣れていないユーザの負担をかなり軽減し得、さらにNWTジョブの記述上のエラーによるジョブの再投入数を低減化し得、無効なジョブ処理を削減する。この結果、ユーザ個々の数値シミュレーション処理におけるジョブ処理時間が短縮され、延いてはシステム使用時間における有効利用性の向上が図れる。

(2) NSシェルやNSコマンドの利用において、システム運用規約に基づいたジョブ処理を可能にし、バッチリクエストの構成法、さらにはジョブの構成法を定め、ジョブ混雑時にも混乱のない数値風洞の利用を実現し得た。

(3) ジョブ処理においてNSシェルやNSコマンドのシステム標準値や省略値等の設定が有効に働き、その結果、システム資源の有効利用を実現可能にする。

```
aoi% nlog  ·····①

   1 : x01+08BVc       2 : x01+08BYx       3 : x01+09BVc       4 : x01+10DIY

SelectJob(Quit, number, Del number..., Rescan, Help) : 4  ·····②
Select job is x01+10DIY
   1. fortc01      2. go01         3. go02         4. go03
   5. lied@X       6. log          7. shell01      8. shell02
   9. shell03     10. shell@X
------------------------------------------------------------------------
Action(Quit, [View], Del, File, Printer, Status, Rescan, Help) [2:x01+10DIY] 2  ·····③


 (  2. go01 の標準印刷出力結果を表示する。  )  ·····④


   1. fortc01      2. go01         3. go02         4. go03
   5. lied@X       6. log          7. shell01      8. shell02
   9. shell03     10. shell@X
------------------------------------------------------------------------
Action(Quit, [View], Del, File, Printer, Status, Rescan, Help) [2:x01+10DIY]: p 2  ·····⑤

Please input Printer name : cntpr1  ·····⑥
要求ＩＤは cntpr1-73498 (標準入力) です
   1. fortc01      2. go01         3. go02         4. go03
   5. lied@X       6. log          7. shell01      8. shell02
   9. shell03     10. shell@X
------------------------------------------------------------------------
Action(Quit, [View], Del, File, Printer, Status, Rescan, Help) [2:x01+10DIY]:
```

図6.8　nlogコマンドの表示結果

7. おわりに

ワークステーションは元より、小規模および大規模の計算機システムのOSとして利用されているUNIXは、近年には非常に身近な存在のパソコンにもフリーソフトウェアのシステムとして参入し、そのユーザ数を激増する一途にある。このようにUNIXの利用が標準的となってきた最中に、超高速かつ大規模システムである数値風洞のジョブ処理において新たなUNIXシステム構築法を示し得た。このシステム構築は一見、ユーザ自由度の非常に高い、かつ使い勝手のよいUNIXを、型にはめた融通の利かないシステムと化したものと捉えられるが、本稿に示すとおり、ユーザおよび運用管理者の両者に有効な運用システムとして、平成8年より実運用を継続している。一方、力のあるUNIXユーザには本来の標準システムを利用する道も残しているが、日々、多数のジョブで非常に混雑するシステム運用状況下において、総処理件数の9割以上を占めるジョブがNSシェルとNSコマンドを利用して発生している。なお、NWTは航技研独自の各種機能が有効に働き、システムの平均処理装置稼働率は90%前後の高い値を呈している。以上から、計算機システムの高度有効利用性を目標とするシステム構築は非常に有用であると判断する。

おわりに当たり、実際の運用システム開発に中心的なご活躍をされた富士通（株）の山口靖、矢澤克己氏に、また、NSシェルおよびNSコマンドの設計にご協力を頂いた藤田信英氏に対して、末筆ながら感謝の意を表する。

8. 参考文献

1) 三好、吉岡、他：“数値風洞のハードウェア”第9回航空機計算空気力学シンポジウム論文集 SP-16 (1991)
2) 福田、末松、他：“数値風洞のオペレーティングシステム”、第9回航空機計算空気力学シンポジウム論文集 SP-16 (1991)
3) 土屋：“数値風洞用MSPビュー・ユーザ・インターフェースの開発”、航技研報告 TR-1405 (2000)
4) FUJITSU ネットワークキューイングシステム説明書
5) 中村、石塚、吉田：“航技研 FACOM230-75 アレイプロセッサシステムセンタルーチンの作成”、航技研資料 TM-344 (1978)
6) 末松、吉田、土屋、畑山：“主記憶有効利用のための諸方策および航技研システムへの適用”、航技研資料 TM-419 (1980)
7) 土屋、末松、吉田、畑山：“計算機システムにおけるジョブ処理用新スケジューラの提案”、航技研報告 TR-659 (1981)
8) 土屋：“NSシステム用ジョブ・ジョブステップ・スケジューラの開発”、航技研報告 TR-977 (1988)
9) 土屋、末松、畑山：“次期航技研システム用ジョブ制御マクロの設計”、航技研資料 TM-444 (1981)
10) 土屋：“NSシステム用光磁気ディスク装置管理用運用プログラムの開発”、航技研報告 TR-1086 (1990)
11) 末松：“数値風洞用ジョブスケジューラの開発”、航技研報告 TR-1383 (1999)

表４．４　ｎｊｏｂ文（NWTジョブの始まりを示す。）

```
njob    [ -y ] [ -m ]  ｼﾞｮﾌﾞｸﾗｽ
```

パラメータ	省略値	パラメータの説明
-y		各バッチリクエストの実行開始／終了をメールで通知する。
-m		実際のサブミットは行わず、標準出力に展開後のスクリプトを出力する。
ｼﾞｮﾌﾞｸﾗｽ		ジョブクラスを指定する（省略不可）。

＜機能＞

(1) ジョブクラスには、ｕ００１～ｕ１６６，ｌ００１～ｌ００４の範囲で指定できる。

(2) ｎｊｏｂ文以下の中間シェルでコメントにする場合には、先頭に『＃』を記述する。

＜使用例＞

(1) １ｐｅのNWTジョブの翻訳・結合・実行を行う。

```
njob u001
nfortc src1.f src2.f
nlied
ngo -t 300
nusdkr -n 10 /large/x/x01/rdata
nusdkw -n 20 /large/x/x01/wdata
```

(2) ６４ｐｅのNWTジョブの翻訳・結合・実行を行う。

```
njob u064
nfortc -p '-Wx' src1.f src2.f
nlied -p '-Wx'
ngo -t 600
nusdkr -n 10 /large/x/x01/rdata
nusdkw -n 20 /large/x/x01/wdata
```

(3) インクルードファイルを参照して、翻訳・結合を行い、ロードモジュールｂ．ｏｕｔを作成する。

```
njob u001
nfortc -p '-I /home/x/x01/incl' src1.f src2.f
nlied -o b.out
```

(4) 翻訳・結合を行い、ユーザライブラリｌｉｂｕｓｒ１．ａを作成する。

```
njob u001
nulib -o /small/usrlib/libusr1.a src1.f src2.f
```

(5) ｖｆｌファイルを利用したジョブの翻訳・結合・実行を行う。

```
njob u001
nfortc src1.f src2.f
nlied
ngo -t 300
nusdkr -n 10 -b 1000 /large/x/x01/rdata
nusdkw -n 20 -i 10 -e 5 -b 1000 /large/x/x01/wdata
```

(6) ユーザライブラリｌｉｂｕｓｒ１．ａを利用したジョブの翻訳・結合・実行を行う。

```
njob u001
nfortc src1.f src2.f
nlied -p '-L /small/usrlib -l usr1'
ngo -t 300
nusdkr -n 10 /large/x/x01/rdata
nusdkw -n 20 /large/x/x01/wdata
```

表４．５　ｎｆｏｒｔｃ文(NWT用ＦＯＲＴＲＡＮプログラムの翻訳を行う。)

```
nfortc　[ -p 'ｺﾝﾊﾟｲﾗｵﾌﾟｼｮﾝ' ]　ｿｰｽﾌｧｲﾙ名 ･･･
```

パラメータ	省略値	パラメータの説明
-p 'ｺﾝﾊﾟｲﾗｵﾌﾟｼｮﾝ'	ｼｽﾃﾑ標準値	ＵＸＰコンパイラオプションを指定する。
ｿｰｽﾌｧｲﾙ名		ソースファイル名を指定する。

＜機能＞
(1) 並列化する場合には、コンパイラオプションに『－Ｗｘ』を指定する。
(2) オウンコンパイラでコンパイルする場合に、以下の言語仕様のコンパイラオプションを指定する。
　　ＦＯＲＴＲＡＮ７７　：　－Ｘｆ７
　　Ｆｏｒｔｒａｎ９０　：　－Ｘ９
(3) ファイルおよびディレクトリの検索は、各ユーザのホームディレクトリ（x01 の場合は、/home/x/x01）を基準にしている。
(4) ソースファイル名は命名規約として、『～.f』でなければならない。
(5) インクルードディレクトリの指定は、コンパイラオプションの中で以下のように指定する。
　　『－Ｉ　インクルードディレクトリ名』
＜使用例＞
(1) ２つのソースファイルを翻訳して並列化する。

```
nfortc -p '-Wx -Ps' src1.f src2.f
```

(2) ディレクトリ配下の全ファイルをインクルードとともに翻訳する。

```
nfortc -p '-Psd -I /small/inc1 -I /small/inc2' /small/src1/*.f
```

(3) オウンコンパイラのＦＯＲＴＲＡＮ７７を利用して翻訳する。

```
nfortc -p '-Xf7 -Ps' src1.f src2.f
```

表４．６　ｎｌｉｅｄ文(NWT用ＦＯＲＴＲＡＮプログラムの結合・編集を行う。)

```
nlied　[ -p 'ﾘﾝｹｰｼﾞｵﾌﾟｼｮﾝ' ]　[ -o ﾛｰﾄﾞﾓｼﾞｭｰﾙﾌｧｲﾙ名 ]
```

パラメータ	省略値	パラメータの説明
-p 'ﾘﾝｹｰｼﾞｵﾌﾟｼｮﾝ'		ＵＸＰリンケージオプションを指定する。
-o ﾛｰﾄﾞﾓｼﾞｭｰﾙﾌｧｲﾙ名	a.out	ロードモジュールを保存するファイル名を指定する。

＜機能＞
(1) 並列化する場合には、リンケージオプションに『－Ｗｘ』を指定する。
(2) ファイルおよびディレクトリの検索は、各ユーザのホームディレクトリ（x01 の場合は、/home/x/x01）を基準にしている。
(3) ユーザライブラリの指定は、リンケージオプションの中で以下のように指定する。なお、指定時に使用するユーザライブラリの部分名とは、『 lib～.a』の『～』の部分を示す。
　　『－Ｌ　ディレクトリ名　－ｌ　ユーザライブラリの部分名』

＜使用例＞
(1) ユーザライブラリとともに結合・編集して並列化する。

```
nlied -p '-Wx -L /small/usrlib1 -l lib1 -L /small/usrlib2 -l lib2'
```

(2) ユーザライブラリとともに結合・編集してロードモジュールを保存する。

```
nlied -p '-L /small/usrlib1 -l lib1' -o /small/module/module1
```

表４．７　ｎｇｏ文（ NWT用ＦＯＲＴＲＡＮプログラムを実行する。）

```
ngo [ -p '実行時ｵﾌﾟｼｮﾝ' ] [ -t 時間 ] [ -c ｶｰﾄﾞﾃﾞｰﾀﾌｧｲﾙ名 ] [ ﾛｰﾄﾞﾓｼﾞｭｰﾙﾌｧｲﾙ名 ]
```

パラメータ	省略値	パラメータの説明
-p '実行時ｵﾌﾟｼｮﾝ'	ｼｽﾃﾑ標準値	実行時オプションを指定する。
-t 時間	600	実行時間を秒単位で指定する。
-c ｶｰﾄﾞﾃﾞｰﾀﾌｧｲﾙ名		実行時に使用するカードデータファイル名を指定する。
ﾛｰﾄﾞﾓｼﾞｭｰﾙﾌｧｲﾙ名		実行するロードモジュール名を指定する。

＜機能＞
(1) ファイルおよびディレクトリの検索は、各ユーザのホームディレクトリ（x01 の場合は、/home/x/x01）を基準にしている。
(2) 実行時オプションは、『－Ｗ１』を必ず指定し、これに続けてサブオプションをカンマで区切って指定する。
(3) 浮動小数点の内部表現形式において、M形式のデータを入出力する場合には、実行時オプションに変換対象のファイル識別番号を指定する。
・全ファイルの場合　：　－Ｃ
・特定ファイルの場合　：　－Ｃｎｎ，－Ｃｎｎ，・・・
＜使用例＞
(1) 実行時にバイナリファイルの入出力変換を行う。

```
ngo -p '-W1,-C10,-C20'
```

(2) 実行時にロードモジュールを使用し、実行時間を１時間で打ち切る。

```
ngo -t 3600 /small/module/module1
```

表４．８　ｎｕｓｄｋｒ文（ NWT実行時に、ＵＸＰファイルを参照する。）

```
nusdkr -n 装置番号 [ -b ﾌｫｰﾄﾗﾝﾊﾞｯﾌｧｻｲｽﾞ ] [ ﾌｧｲﾙ名 ]
```

パラメータ	省略値	パラメータの説明
-n 装置番号		プログラム内入出力文の装置番号を指定する。
-b ﾌｫｰﾄﾗﾝﾊﾞｯﾌｧｻｲｽﾞ	ｼｽﾃﾑ標準値	実行時にフォートランの入出力バッファサイズを指定する。
ﾌｧｲﾙ名		実行時に使用するｖｆｌファイル名を絶対パス名で指定する。 実行時に実際の入出力動作が不要の場合には省略する。

＜機能＞
(1) ファイルは既存のｖｆｌファイルに限る。
(2) ファイルの入出力量が大きい場合には、フォートランの入出力バッファサイズを指定する。但し、指定した入出力バッファサイズがメモリに影響するので、プログラムメモリサイズに十分な余裕が必要である。
・順次入出力ファイルの場合　：　ＫＢ単位で指定
・直接入出力ファイルの場合　：　ＦＯＲＴＲＡＮ記録の個数を指定

＜使用例＞
(1) 実行時にｖｆｌファイルを参照する。

```
nusdkr  -n 10  /large/x/x01/data1
```

(2) 実行時にｖｆｌ順次入出力ファイルを入出力バッファサイズを指定して参照する。

```
nusdkr  -n 10  -b 1024  /large/x/x01/data1
```

表４．９　ｎｕｓｄｋｒｗ文(ＮＷＴ実行時に、ＵＸＰファイルを参照・更新する。)

```
nusdkrw    -n 装置番号  [ -b ﾌｫｰﾄﾗﾝﾊﾞｯﾌｧｻｲｽﾞ ]  [ ﾌｧｲﾙ名 ]
```

パラメータ	省略値	パラメータの説明
-n 装置番号		プログラム内入出力文の装置番号を指定する。
-b ﾌｫｰﾄﾗﾝﾊﾞｯﾌｧｻｲｽﾞ	ｼｽﾃﾑ標準値	実行時にフォートランの入出力バッファサイズをＫＢ単位で指定する。
ﾌｧｲﾙ名		実行時に使用するファイル名を絶対パス名で指定する。 実行時に実際の入出力動作が不要の場合には省略する。

＜機能＞
(1) ファイルは既存のｖｆｌファイルに限る。
(2) ファイルの入出力量が大きい場合には、フォートランの入出力バッファサイズを指定する。但し、指定した入出力バッファサイズがメモリに影響するので、プログラムメモリサイズに十分な余裕が必要である。
・順次入出力ファイルの場合　：　ＫＢ単位で指定
・直接入出力ファイルの場合　：　ＦＯＲＴＲＡＮ記録の個数を指定

＜使用例＞
(1) 実行時にｖｆｌファイルを参照・更新する。

```
nusdkrw  -n 20  /large/x/x01/data2
```

(2) 実行時にｖｆｌ順次入出力ファイルを入出力バッファサイズを指定して参照・更新する。

```
nusdkrw  -n 20  -b 1024  /large/x/x01/data2
```

表４．１０　ｎｕｓｄｋｗ文(ＮＷＴ実行時に、ＵＸＰファイルを作成・更新する。)

```
nusdkw    -n 装置番号  [ -i 初期値 ]  [ -e 増分値 ]  [ -b ﾌｫｰﾄﾗﾝﾊﾞｯﾌｧｻｲｽﾞ ]  [ ﾌｧｲﾙ名 ]
```

パラメータ	省略値	パラメータの説明
-n 装置番号		プログラム内入出力文の装置番号を指定する。
-i 初期値	5	新規に作成するファイルがｖｆｌファイルの場合にブロック単位で指定する。
-e 増分値	3	新規に作成するファイルがｖｆｌファイルの場合にブロック単位で指定する。
-b ﾌｫｰﾄﾗﾝﾊﾞｯﾌｧｻｲｽﾞ	ｼｽﾃﾑ標準値	実行時にフォートランの入出力バッファサイズを指定する。
ﾌｧｲﾙ名		実行時に使用するファイル名を絶対パス名で指定する。 実行時に実際の入出力動作が不要の場合には省略する。

＜機能＞
(1) 新規に作成するファイルがｖｆｌファイルの場合に、ｖｆｌファイルがディレクトリの配下によって、１ブロックあたりの単位を各々で設定してあるので容量に注意すること。
・／ａｒｒａｙ　：　８ＭＢ（１ブロック）
・／ｌａｒｇｅ　：　６００ＫＢ（１ブロック）
・／ｌｇｗｏｒｋ　：　６００ＫＢ（１ブロック）

(2) ファイルの入出力量が大きい場合には、フォートランの入出力バッファサイズを指定する。但し、指定した入出力バッファサイズがメモリに影響するので、プログラムメモリサイズに十分な余裕が必要である。
・順次入出力ファイルの場合 ： ＫＢ単位で指定
・直接入出力ファイルの場合 ： ＦＯＲＴＲＡＮ記録の個数を指定

＜使用例＞

(1) 実行時にｖｆｌファイルを更新する。

```
nusdkw  -n 30  /large/x/x01/data3
```

(2) 実行時にｖｆｌ順次入出力ファイルを入出力バッファサイズを指定して更新する。

```
nusdkw  -n 30  -b 1024  /large/x/x01/data3
```

(3) 実行時にｖｆｌファイルを新規に作成する。

```
nusdkw  -n 30  -i 100 -e 50  /large/x/x01/data3
```

表４．１１　ｎｘｙ文(ＮＷＴ実行時に、ＸＹプロッタイメージの図形データを出力する。)

```
nxy      [ -i 初期値 ] [ -e 増分値 ] [ -b ﾌｫｰﾄﾗﾝﾊﾞｯﾌｧｻｲｽﾞ ] [ ﾌｧｲﾙ名 ]
```

パラメータ	省略値	パラメータの説明
-i 初期値	5	新規に作成するファイルがｖｆｌファイルの場合にブロック単位で指定する。
-e 増分値	3	新規に作成するファイルがｖｆｌファイルの場合にブロック単位で指定する。
-b ﾌｫｰﾄﾗﾝﾊﾞｯﾌｧｻｲｽﾞ	ｼｽﾃﾑ標準値	実行時にフォートランの入出力バッファサイズを指定する。
ﾌｧｲﾙ名		実行時に使用するファイル名を絶対パス名で指定する。 実行時に実際の入出力動作が不要の場合には省略する。

＜機能＞

(1) 新規に作成するファイルがｖｆｌファイルの場合に、ｖｆｌファイルがディレクトリの配下によって、１ブロックあたりの単位を各々で設定してあるので容量に注意すること。
・／ａｒｒａｙ　： ８０００ＫＢ（１ブロック）
・／ｌａｒｇｅ　： ６００ＫＢ（１ブロック）
・／ｌｇｗｏｒｋ　： ６００ＫＢ（１ブロック）

(2) ファイルの入出力量が大きい場合には、フォートランの入出力バッファサイズを指定する。但し、指定した入出力バッファサイズがメモリに影響するので、プログラムメモリサイズに十分な余裕が必要である。
・順次入出力ファイルの場合 ： ＫＢ単位で指定
・直接入出力ファイルの場合 ： ＦＯＲＴＲＡＮ記録の個数を指定

＜使用例＞

(1) 実行時に図形データを既存のｖｆｌファイルに出力する。

```
nxy  /large/x/x01/xy1
```

(2) 実行時に図形データを新規にｖｆｌファイルに出力する。

```
nxy  -i 10 -e 10  /large/x/x01/xy2
```

表４．１２　ｎｕｌｉｂ文（ＮＷＴ用自動呼び出しライブラリを作成する。）

nulib　[-p 'ｺﾝﾊﾟｲﾗｵﾌﾟｼｮﾝ'] -o ﾕｰｻﾞﾗｲﾌﾞﾗﾘ名　ｿｰｽﾌｧｲﾙ名 ･･･

パラメータ	省略値	パラメータの説明
-p 'ｺﾝﾊﾟｲﾗｵﾌﾟｼｮﾝ'		コンパイラオプションを指定する。
-o ﾕｰｻﾞﾗｲﾌﾞﾗﾘ名		ユーザライブラリを保存するファイル名を指定する。
ｿｰｽﾌｧｲﾙ名		ソースファイル名を指定する。

＜機能＞

(1) オウンコンパイラでコンパイルする場合に、以下の言語仕様のコンパイラオプションを指定する。

ＦＯＲＴＲＡＮ７７　：　－Ｘｆ７

Ｆｏｒｔｒａｎ９０　：　－Ｘ９

(2) ファイルおよびディレクトリの検索は、各ユーザのホームディレクトリ（x01の場合は、/home/x/x01）を基準にしている。

(3) ユーザライブラリ名は命名規約として、『 lib～.a』でなければならない。

(4) ソースファイル名は命名規約として、『～.f』でなければならない。

(5) インクルードディレクトリの指定は、コンパイラオプションの中で以下のように指定する。

『－Ｉ　インクルードディレクトリ名』

＜使用例＞

(1) ２つのソースファイルを翻訳してライブラリを作成する。

```
nulib  -o /small/usrlib/lib1  src1.f  src2.f
```

(2) ディレクトリ配下の全ファイルをインクルードとともに翻訳してライブラリを作成する。

```
nulib  -p '-I /small/inc1 -I /small/inc2'  -o /small/usrlib/lib2  /small/src2/*.f
```

表４．１３　ａｃｔｊｏｂ（ＮＷＴジョブの処理状況を表示する。）

コマンド名	オ　ペ　ラ　ン　ド
actjob	なし

＜機能＞

ＮＷＴジョブの処理状況を表示する。

表４．１４　ａｌｌｏｃ（ｖｆｌファイルを割り当てる。）

コマンド名	オ　ペ　ラ　ン　ド
alloc	[-f]　初期量　増分量　vfl ﾌｧｲﾙ名

＜機能＞

ｖｆｌファイルを割り当てる。

＜オペランドの説明＞

(1) -f

既存のｖｆｌファイルに対して内容を破棄し、再割り当てをする場合に指定する。

(2) 初期量　増分量

割り当てするファイルの容量を指定する。

・容量ｋ　：ＫＢ単位

・容量ｍ　：ＭＢ単位

・容量　　：ブロック単位でディレクトリによって単位を設定

／ａｒｒａｙ　････８０００ＫＢ

／ｌａｒｇｅ　････　６００ＫＢ

／ｌｇｗｏｒｋ････　６００ＫＢ

(3) vfl ﾌｧｲﾙ名
　　ｖｆｌファイル名を指定する。ｖｆｌファイル以外を指定するとエラーとなる。
＜入力例＞
(1) ｘ０１ユーザが、/large配下に初期量５ＭＢ、増分量３ＭＢ単位で、ｖｆｌファイルを新規に割り当てる。

```
alloc  5m  3m  /large/x/x01/testdata
```

表４．１５　ｆ７７ｎｗｔ（ＮＷＴ用ロードモジュールを作成する。）

コマンド名	オ　ペ　ラ　ン　ド
f77nwt	［ frtpx ｵﾌﾟｼｮﾝ ］［ -o ﾛｰﾄﾞﾓｼﾞｭｰﾙ名 ］ ｿｰｽﾌｧｲﾙ名 または ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙ名 ・・・

＜機能＞
　ＮＷＴ用ロードモジュールを作成する。
＜オペランドの説明＞
(1) frtpx ｵﾌﾟｼｮﾝ
　　ＵＸＰコンパイルオプション等を指定する。詳細については、『ｍａｎ　ｆｒｔｐｘ』にて参照できる。
　　更に、以下のオプションも指定して利用することができる。
　　-Udc ： 倍精度用 calcomp ライブラリ利用
　　-Umpi ： MPI ライブラリ利用
　　-Upvm ： PVM ライブラリ利用
(2) -o ﾛｰﾄﾞﾓｼﾞｭｰﾙ名
　　保存するロードモジュール名を指定する。省略した場合には、ロードモジュール名『a.out』に出力する。
(3) ｿｰｽﾌｧｲﾙ名 または ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙ名 ・・・
　　ソースファイル名またはオブジェクトファイル名を指定する。
＜特記事項＞
(1) ソースファイル名は命名規約として、『～.f』でなければならない。
(2) オブジェクトファイル名は命名規約として、『～.o』でなければならない。
＜入力例＞
(1) ソースファイルを使用して標準最適化されたＮＷＴ用ロードモジュールを作成する。

```
f77nwt  -Wx  -Psd  -o module5  src1.f  src2.f
```

表４．１６　ｎｃａｎ（ＮＷＴジョブをキャンセルする。）

コマンド名	オ　ペ　ラ　ン　ド
ncan	NWT ｼﾞｮﾌﾞ名

＜機能＞
　ＮＷＴジョブをキャンセルする。
＜オペランドの説明＞
(1) NWT ｼﾞｮﾌﾞ名
　　ＮＷＴのジョブ名を指定する。
＜特記事項＞
(1) ＭＳＰから投入されたＮＷＴジョブはキャンセルすることができない。
＜入力例＞
(1) ｘ０１＋３１Ａｂ１のＮＷＴジョブをキャンセルする。

```
ncan  x01+31Ab1
```

表4．17　ｎｌｏｇ（NWTジョブの処理結果を検索する。）

コマンド名	オ　ペ　ラ　ン　ド
nlog	[NWTｼﾞｮﾌﾞ名]

＜機能＞
NWTジョブの処理結果を検索する。
＜オペランドの説明＞
(1) NWTｼﾞｮﾌﾞ名
検索を行うNWTのジョブ名を指定する。本オペランドを省略すると、ジョブ名の一覧を表示する。
＜入力例＞
(1) ｘ０１＋３１Ａｂ１のジョブの処理結果を見る。

```
nlog  x01+31Ab1
```

表4．18　ｎｑｕｏｔａ（ユーザファイルの割当量および使用量を表示する。）

コマンド名	オ　ペ　ラ　ン　ド
nquota	[-v]

＜機能＞
ユーザファイルの割当量および使用量を表示する。また、マイグレーション情報も表示する。
＜オペランドの説明＞
(1) -v
対象となるすべてのファイルシステムについて割当量および使用量を表示する。本オペランドを省略すると、制限を越えているときに限り、情報が表示される。
＜入力例＞
(1) 割当量および使用量を表示する。

```
nquota  -v
```

表4．19　ｎｓ（セッションユーザのNWTジョブの処理状況を表示する。）

コマンド名	オ　ペ　ラ　ン　ド
ns	なし

＜機能＞
セッションユーザのNWTジョブの処理状況を表示する。

表4．20　ｎｓｕｂ（クロスコンパイラ使用NWTジョブを投入する。）

コマンド名	オ　ペ　ラ　ン　ド
nsub	[-g ｸﾞﾙｰﾌﾟ名] [-m] 中間ｼｪﾙﾌｧｲﾙ名

＜機能＞
クロスコンパイラを使用するNWTジョブを投入する。
＜オペランドの説明＞
(1) -g ｸﾞﾙｰﾌﾟ名
処理する当該グループ名を指定する。
(2) -m
実際の投入は行ず、標準出力に展開後のスクリプトを出力する。
(3) 中間ｼｪﾙﾌｧｲﾙ名
処理を記述した中間ｼｪﾙﾌｧｲﾙ名を指定する。
＜入力例＞
(1) クロスコンパイラを使用するNWTジョブを、グループｇｒｐ１で中間シェルファイルを利用して実行する。

```
nsub  -g grp1  sh1
```

表4．21　ｎｓｕｂｏ（オウンコンパイラ使用NWTジョブを投入する。）

コマンド名	オ　ペ　ラ　ン　ド
nsubo	［ -g ｸﾞﾙｰﾌﾟ名 ］［ -m ］　中間ｼｪﾙﾌｧｲﾙ名

＜機能＞
　オウンコンパイラを使用するＮＷＴジョブを投入する。
＜オペランドの説明＞
(1) -g ｸﾞﾙｰﾌﾟ名
　　処理する当該グループ名を指定する。
(2) -m
　　実際の投入は行ず、標準出力に展開後のスクリプトを出力する。
(3) 中間ｼｪﾙﾌｧｲﾙ名
　　処理を記述した中間ｼｪﾙﾌｧｲﾙ名を指定する。
＜入力例＞
(1) オウンコンパイラを使用するＮＷＴジョブを、グループｇｒｐ１で中間シェルファイルを利用して実行する場合の展開後のスクリプトを表示する。

```
nsubo  -g grp1  -m  sh1
```

表4．22　ｎｕｌｉｂ（ＮＷＴ用ユーザライブラリを作成する。）

コマンド名	オ　ペ　ラ　ン　ド
nulib	［ -p 'ｺﾝﾊﾟｲﾗｵﾌﾟｼｮﾝ ･･･' ］ -o ﾕｰｻﾞﾗｲﾌﾞﾗﾘ名　ｿｰｽﾌｧｲﾙ名 ･･･

＜機能＞
　ＮＷＴ用ユーザライブラリを作成する。
＜オペランドの説明＞
(1) -p 'ｺﾝﾊﾟｲﾗｵﾌﾟｼｮﾝ ･･･'
　　ＵＸＰコンパイラオプションを指定する。
(2) -o ﾕｰｻﾞﾗｲﾌﾞﾗﾘ名
　　作成するユーザライブラリ名を指定する。
(3) ｿｰｽﾌｧｲﾙ名 ･･･
　　ソースプログラムのファイル名を指定する。
＜特記事項＞
(1) ソースファイル名は命名規約として、『～.f』でなければならない。
(2) ユーザライブラリ名は命名規約として、『 lib～.a』でなければならない。
＜入力例＞
(1) ソースファイルを使用して、ＮＷＴ用ユーザライブラリを作成する。

```
nulib  -p '-Oe'  -o libusr1.a  src1.f  src2.f
```

表4．23　ｒｌｓｅ（ｖｆｌファイルの未使用領域を開放する。）

コマンド名	オ　ペ　ラ　ン　ド
rlse	vfl ﾌｧｲﾙ名

＜機能＞
　ｖｆｌファイルの未使用領域を開放する。
＜オペランドの説明＞
(1) vfl ﾌｧｲﾙ名
　　未使用領域を開放するｖｆｌファイル名を指定する。
＜特記事項＞
(1) ｖｆｌファイル以外のファイル名を指定した場合は、エラーとなる。
(2) ファイルの所有者または現グループに位置しているユーザでなければ開放できない。
(3) 使用中のファイルは開放できない。

(4) 開放中に当該ファイルを使用した場合には動作は保証されないので注意すること。

＜入力例＞

(1) ディレクトリ/large配下のｖｆｌファイルの未使用領域を開放する。

```
rlse  /large/x/x01/testdata
```

表4．24　ｖｆｌｒｍ(ｖｆｌファイルを削除する。)

コマンド名	オ　ペ　ラ　ン　ド
vflrm	vfl ﾌｧｲﾙ名 ･･･

＜機能＞

ｖｆｌファイルを使用状態の判定に従って削除する。

＜オペランドの説明＞

(1) vfl ﾌｧｲﾙ名 ･･･

判定するｖｆｌファイル名を指定する。

＜特記事項＞

(1) ｖｆｌファイル以外のファイル名を指定した場合は、エラーとなる。

＜入力例＞

(1) ディレクトリ/large配下のｖｆｌファイルを削除する。

```
vflrm  /large/x/x01/testdata
```

表4．25　ｖｆｌｕｓｅ(ｖｆｌファイルの使用状態を判定する。)

コマンド名	オ　ペ　ラ　ン　ド
vfluse	vfl ﾌｧｲﾙ名 ･･･

＜機能＞

ｖｆｌファイルがプロセスによって使用状態であるかを判定する。

＜オペランドの説明＞

(1) vfl ﾌｧｲﾙ名 ･･･

判定するｖｆｌファイル名を指定する。

＜特記事項＞

(1) ｖｆｌファイル以外のファイル名を指定した場合は、エラーとなる。

＜入力例＞

(1) ディレクトリ/large配下のｖｆｌファイルの使用状態を表示する。

```
vfluse  /large/x/x01/testdata
```

表4．26　ｗａｉｔｊｏｂ(NWTジョブの処理待ち状況を表示する。)

コマンド名	オ　ペ　ラ　ン　ド
waitjob	なし

＜機能＞

NWTジョブの処理待ち状況を表示する。

航空宇宙技術研究所報告1410号

平成12年8月発行

発行所　科学技術庁航空宇宙技術研究所
東京都調布市深大寺東町7-44-1
電話（0422）40-3075　〒182-8522
印刷所　株式会社　実業公報社
東京都千代田区九段北1-7-8

Printed in Japan