Sun™ Mainframe Batch Manager ソフトウェア リファレンスマニュアル

Release 10.1.0

Sun Microsystems, Inc.
www.sun.com

Part No. 819-2360-10
2005 年 6 月, Revision A

コメントの送付: http://www.sun.com/hwdocs/feedback

原典: *Sun™ Mainframe Batch Manager Software Reference Guide*
Part No: 817-7444-10
Revision A

目次

表目次

はじめに

このマニュアルでは、MVS および VSE Job Control Language (JCL) についての参照情報、および Sun™ Mainframe Batch Manager (Sun MBM) トランスレータによって生成されるマクロ文についての参照情報を提供します。

マニュアルの構成

このマニュアルは、以下の章で構成されています。

第 1 章では、サポートされるおよびサポートされない MVS JCL 文についての情報を提供します。

第 2 章では、サポートされるおよびサポートされない VSE JCL 文についての情報を提供します。

第 3 章では、Sun MBM マクロ文について説明し、マクロによるジョブおよびプロシージャーの作成についての情報を提供します。

付録 A では、特殊文字に関して、翻訳時の変換規則およびマクロ文でのコーディング法についての情報を提供します。

UNIX コマンド

このマニュアルには、システムの停止、システムの起動、およびデバイスの構成などの基本的な UNIX® コマンドと操作手順に関する説明はありません。これらについては、以下を参照してください。

- ご使用のシステムに付属のソフトウェアマニュアル
- 下記にある Solaris™ オペレーティングシステムのマニュアル

  `http://docs.sun.com`

シェルプロンプトについて

シェル	プロンプト
UNIX の C シェル	**マシン名**`%`
UNIX の Bourne シェルと Korn シェル	`$`
スーパーユーザー (シェルの種類を問わない)	`#`

書体と記号について

書体または記号*	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例。	`.login` ファイルを編集します。 `ls -a` を実行します。 `% You have mail.`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表します。	`%` **`su`** `Password:`
AaBbCc123 または **ゴシック**	コマンド行の可変部分。実際の名前や値と置き換えてください。	`rm` *filename* と入力します。 `rm` **ファイル名** と入力します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』

書体または記号*	意味	例
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照。 この操作ができるのは「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅をこえる場合に、継続を示します。	% **grep '^#define ** **XV_VERSION_STRING'**
[ ]	省略可能な項目を示します。	dltjob *jon* [-n *name*]
{ }	互いに排他的な必須引数を囲みます。	ONCONDCODE *comparator integer* [*operator comparator integer*] {BYPASS | CONTINUE | GOTO {*label-name* | *stepname* | END_JOB}} [scope= JOB | STEP] [verbose][poverride='y' stepname='*stepname*']
|	区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。	abtjob *jon* [-s job | cmd]

* 使用しているブラウザにより、これら設定と異なって表示される場合があります。

このマニュアルでは、次の書式を使用してコマンドを表記します。

```
$ command required-argument [optional-argument]
```

コマンドに省略可能な引数が記述されていない場合は、そのコマンドを入力して Return キーを押します。

ファイル識別子

ファイル識別子は、次の 2 つの部分から構成されます。

- ディレクトリ、または 1 つ以上のディレクトリを指定できる環境変数
- ファイル識別子の最後の構成要素であるファイル名

ファイル識別子	説明
ディレクトリ	Sun MBM で使用される絶対ディレクトリ名は、60 文字以内でなければなりません。パス名の任意の部分に代えて、先頭にドル符号 ($) を付けた環境変数を使用できます。たとえば、次の 2 行はどちらも有効であり、同じディレクトリを指定します。 • `/local/mbm/pack/bin` • `$PACK/bin` 2 行目の `PACK` 環境変数は `/local/mbm/pack` に設定されています。`$` 指示子を使用することにより、`PACK` 環境変数がそれの完全な意味に展開されます。
環境変数	ディレクトリ名やファイル名を含む名前であり、通常 1 ～ 14 個の大文字です。
ファイル名	ファイル名は、ご使用のプラットフォームの制限事項に従う必要があります。

関連マニュアル

製品	タイトル	Part No.
Sun Mainframe Batch Manager ソフトウェア	『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』	819-2505-10
	『Sun Mainframe Batch Manager ソフトウェア インストールガイド』	819-2506-10
	『Sun Mainframe Batch Manager ソフトウェア メッセージガイド』	819-2507-10
	『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』	819-2508-10
	『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』	819-2509-10
	『Sun Mainframe Batch Manager ソフトウェア ご使用にあたって (Solaris プラットフォーム用)』	819-2510-10
	『Sun Mainframe Batch Manager ソフトウェア 高可用性 (HA) データサービス (Sun Cluster 用)』	819-2511-10

製品	タイトル	Part No.
Sun Mainframe Transaction Processing ソフトウェア	『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』	819-2514-10
	『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』	819-2515-10
	『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』	819-2516-10
	『Sun Mainframe Transaction Processing ソフトウェア インストールガイド』	819-2517-10
	『Sun Mainframe Transaction Processing ソフトウェア メッセージガイド』	819-2518-10
	『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』	819-2519-10
	『Sun Mainframe Transaction Processing ソフトウェア 障害追跡とチューニング』	819-2520-10
	『Sun Mainframe Transaction Processing ソフトウェア ご使用にあたって (Solaris プラットフォーム用)』	819-2521-10
	『Sun Mainframe Transaction Processing ソフトウェア 高可用性 (HA) データサービス (Sun Cluster 用)』	819-2522-10
IBM MVS	『IBM MVS/ESA JCL Reference』	GC28-1479
IBM VSE	『IBM VSE/ESA System Control Statements』	SC33-6713
	『IBM VSE/POWER Administration and Operation』	SC33-6733
Micro Focus Server Express	Micro Focus Server Express マニュアル CD-ROM	*
Acucorp ACUCOBOL-GT	Acucorp ACUCOBOL-GT のマニュアル	*
Liant Open PL/I	『Liant Open PL/I User's Guide』	*
	『Liant Open PL/I Language Reference Manual』	*
	『Liant CodeWatch Reference Manual』	*
C	C コンパイラのマニュアル	*

* これらのマニュアルは、ご使用のプラットフォームによって異なります。プラットフォームに該当するマニュアルについては、ご購入先にお問い合わせください。

Sun のオンラインマニュアル

各言語対応版を含む Sun の各種マニュアルは、次の URL から表示または印刷、購入できます。

http://www.sun.com/documentation

Sun の技術サポート

このマニュアルに記載されていない技術的な問い合わせについては、次の URL にアクセスしてください。

http://www.sun.com/service/contacting

コメントをお寄せください

弊社では、マニュアルの改善に努力しており、お客様からのコメントおよびご忠告をお受けしております。コメントは下記よりお送りください。

http://www.sun.com/hwdocs/feedback

コメントには下記のタイトルと Part No. を記載してください。

『Sun Mainframe Batch Manager ソフトウェア リファレンスマニュアル』、
Part No. 819-2360-10

第1章

MVS JCL

この章の内容は、次のとおりです。

各文の構文についての説明は、IBM MVS マニュアルを参照してください。

変換例については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

MVS JCL 文のサポート

MVS JCL 文の多くはメインフレーム固有であり、Sun MBM 環境には適用されません。これらの文はトランスレータによって変換されず、変換プロセス時に警告が発せられます。そのメッセージについては、『Sun Mainframe Batch Manager ソフトウェア メッセージガイド』を参照してください。

サポートされる MVS JCL 文

次の表は、Sun MBM がサポートする MVS JCL 文の一覧を表示します。サポートされる JES2 文のリストは、表 1-3 を参照してください。

表 1-1 サポートされる MVS JCL 文 (1 / 7)

文	パラメータ	使用法	トランスレータのアクション
//* *comment*		コメントを挿入する	シェルコメント行を作成する
/*		インストリームデータの区切り記号	

表 1-1　サポートされる MVS JCL 文 (2 / 7)

文	パラメータ	使用法	トランスレータのアクション
//		ジョブまたはプロシージャーの終了を指定する	ENDJOB マクロを生成する
DD		データセットを定義および記述する	
DD	*		ASSGNDD マクロを生成する
DD	DATA		ASSGNDD マクロを生成する
DD	DUMMY		type='DUMMY' の ASSGNDD マクロを生成する
DD	DYNAM		無視される
DD	ACCODE		無視される
DD	AMP		無視される
DD	AVGREC		無視される
DD	BURST		burst=*value* パラメータがある SETPRINT マクロを生成する
DD	CHARS		chars=*value* パラメータがある SETPRINT マクロを生成する
DD	CHKPT		無視される
DD	CNTL		無視される
DD	COPIES		copies=*value* パラメータがある SETPRINT マクロを生成する
DD	DATACLAS		無視される
DD	DCB		LRECL および RECFM サブパラメータ用にサポートされる
DD	DDNAME		部分的にサポートされる
DD	DEST		dest=*value* パラメータがある SETPRINT マクロを生成する
DD	DISP		該当するパラメータがある ASSGNDD マクロを生成する
DD	DLM	DD 文の DLM= パラメータにより定義される 2 文字のインストリームデータの区切り記号	type='INSTREAM' パラメータがある ASSGNDD マクロを生成する
DD	DSID		無視される
DD	DSNAME		ASSGNDD マクロを生成する
DD	EXPDT		無視される

表 1-1　サポートされる MVS JCL 文 (3 / 7)

文	パラメータ	使用法	トランスレータのアクション
DD	FCB		fcb=*value* パラメータがある SETPRINT マクロを生成する
DD	FLASH		flash=*value* パラメータがある SETPRINT マクロを生成する
DD	FREE		無視される
DD	HOLD		hold=*value* パラメータがある SETPRINT マクロを生成する
DD	KEYLEN		無視される
DD	KEYOFF		無視される
DD	LABEL		無視される
DD	LIKE		無視される
DD	LRECL		recsize=*value* パラメータがある ASSGNDD マクロを生成する
DD	MGMTCLAS		無視される
DD	MODIFY		modify=*value* パラメータがある SETPRINT マクロを生成する
DD	MSVGP		無視される
DD	OUTLIM		outlim=*value* パラメータがある SETPRINT マクロを生成する
DD	OUTPUT		SETPRINT マクロを生成する
DD	PROTECT		無視される
DD	QNAME		無視される
DD	RECFM		recfmt=*value* パラメータがある ASSGNDD マクロを生成する
DD	RECORG		無視される
DD	REFDD		無視される
DD	RETPD		無視される
DD	RLS		rls=*value* パラメータがある ASSGNDD マクロを生成する
DD	SECMODEL		無視される
DD	SPACE		無視される
DD	SUBSYS		無視される

表 1-1　サポートされる MVS JCL 文 (4 / 7)

文	パラメータ	使用法	トランスレータのアクション
DD	SYSOUT		ASSGNDD マクロを生成するまた、DD SYSOUT 文に、ライター名または形式コードサブパラメータのいずれかがコーディングされている場合、SETPRINTマクロを生成する
DD	TERM		無視される
DD	UCS		ucs=*value* パラメータがある SETPRINT マクロを生成する
DD	UNIT		無視される
DD	VOLUME		無視される
DD	JOBCAT DD		LIBDEF マクロを生成する
DD	STEPCAT DD		LIBDEF マクロを生成する
DD	JOBLIB DD		LIBDEF マクロを生成する
DD	STEPLIB DD		LIBDEF マクロを生成する
DD	SYSIN		ASSGNDD マクロを生成する
EXEC	実行	ジョブステップの開始	
EXEC	PGM		ジョブステップの開始を指定する LABEL 文が先行する EXECPGM
EXEC	PROC		ジョブステップの開始を指定する LABEL 文が先行する EXECPRPC マクロを生成する
EXEC	ACCT		無視される
EXEC	ADDRSPC		無視される
EXEC	COND		EVEN および ONLY 条件に対する ONCONDCODE マクロ、およびその他すべての条件に対する ONRETCODE マクロを生成する
EXEC	DPRTY		無視される
EXEC	DYNAMBR		無視される
EXEC	PARM		EXECPGM または EXECPROC マクロで parm=*value* を生成する
EXEC	PERFORM		無視される
EXEC	RD		無視される
EXEC	REGION		無視される
EXEC	TIME		無視される

表 1-1　サポートされる MVS JCL 文 (5 / 7)

文	パラメータ	使用法	トランスレータのアクション
IF ... THEN ELSE ENDIF	条件文	JCL 文の条件付き実行を使用可能にする	IF/THEN/ELSE/ENDIF マクロを生成する
INCLUDE	組み込み	JCLLIB JCL 文によって定義される区分化データセット (PDS) から JCL を組み込む	INCLUDE 文をそれがポイントする JCL に置換する
JCLLIB	JCL ライブラリ	INCLUDE 文によって JCL ストリームにコピー可能な JCL 行を含むメンバーがある PDS を定義する	LIBDEF マクロを生成する
JOB	ジョブ	ジョブの開始を指定する	BEGINJOB マクロを生成する
JOB	アカウンティング情報		サブパラメータ (*sysout-lines*、*cards*、*forms*、*copies*、*line-count*) に対して SETPRINT マクロを生成する
JOB	ジョブ所有者		無視される
JOB	ADDRSPC		無視される
JOB	CLASS		無視される
JOB	COND		ONRETCODE マクロを生成する
JOB	GROUP		無視される
JOB	MSGCLASS		jobclass=*value* パラメータがある SETPRINT マクロを生成する
JOB	MSGLEVEL		無視される
JOB	NOTIFY		無視される
JOB	PASSWORD		無視される
JOB	PERFORM		無視される
JOB	PRTY		無視される
JOB	RD		無視される
JOB	REGION		無視される
JOB	RESTART		無視される
JOB	TIME		無視される
JOB	TYPRUN		無視される
JOB	USER		無視される
OUTPUT	プロシージャー終了	sysout データセットに対する出力関連オプションを定義する	

表 1-1　サポートされる MVS JCL 文 (6 / 7)

文	パラメータ	使用法	トランスレータのアクション
OUTPUT	BURST		burst=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	CHARS		chars=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	CKPTLINE		ckptline=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	CKPTPAGE		ckptpage=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	CKPTSEC		ckptsec=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	CLASS		class=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	COMPACT		compact=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	CONTROL		control=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	COPIES		copies=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	DATACK		datack=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	DEFAULT		JCL で DEFAULT=Y の場合、SETPRINT マクロを生成し、scope パラメータに _DEFAULT を追加する
OUTPUT	DEST		dest=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	FCB		fcb=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	FLASH		flash=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	FORMDEF		formdef=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	FORMS		forms=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	GROUPID		groupid=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	INDEX		index=*value* パラメータがある SETPRINT マクロを生成する

表 1-1　サポートされる MVS JCL 文 (7 / 7)

文	パラメータ	使用法	トランスレータのアクション
OUTPUT	JESDS		jesds=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	LINDEX		lindex=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	LINECT		linect=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	MODIFY		modify=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	PAGEDEF		pagedef=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	PIMSG		pimsg=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	PRMODE		prmode=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	PRTY		prty=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	THRESHLD		thrshld=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	TRC		trc=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	UCS		ucs=*value* パラメータがある SETPRINT マクロを生成する
OUTPUT	WRITER		writer=*value* パラメータがある SETPRINT マクロを生成する
PEND	プロシージャー終了	プロシージャーの終了を特定する	カタログ化プロシージャーの場合、ENDPROC マクロを生成する
PROC	プロシージャー	プロシージャーの開始。インストリームプロシージャーはサポートされない	BEGINPROC マクロを生成する
SET	記号の設定	記号パラメータに値を割り当てる	SETPARM マクロを生成する

サポートされない MVS JCL 文

次の表は、サポートされない MVS JCL 文のリストの一部です。

表 1-2　サポートされない MVS JCL 文

文	使用法	理由
`// command`	MVS 演算子コマンドを入力する	サポートされていないため
`CNTL`	制御文の開始	プログラム制御文は IBM 出力サブシステムに関連し、Sun MBM 環境には適用されないため
`ENDCNTL`	制御文の終了	プログラム制御文は IBM 出力サブシステムに関連し、Sun MBM 環境には適用されないため
`XMIT`	入力ストリームレコードを転送する	サポートされていないため

JES 文のサポート

Sun MBM の現在のリリースでは、JES2 /*OUTPUT、/*JOBPARM、および /*ROUTE 文がサポートされます。その他のすべての JES2 および JES3 JCL 文はサポートされずに無視されます。

次の表は、サポートされる JES2 文の一覧を表示します。

表 1-3 サポートされる JES2 文 (1 / 3)

文	パラメータ	使用法	トランスレータのアクション
/*JOBPARM		JES2 ジョブ関連パラメータを定義する	SETPRINT マクロを生成する
/*JOBPARM	BURST		burst=*value* パラメータがある SETPRINT マクロを生成する
/*JOBPARM	BYTES		bytes=*value* パラメータがある SETPRINT マクロを生成する
/*JOBPARM	CARDS		cards=*value* パラメータがある SETPRINT マクロを生成する
/*JOBPARM	COPIES		copies=*value* パラメータがある SETPRINT マクロを生成する
/*JOBPARM	FORMS		forms=*value* パラメータがある SETPRINT マクロを生成する
/*JOBPARM	LINECT		linect=*value* パラメータがある SETPRINT マクロを生成する
/*JOBPARM	LINES		lines=*value* パラメータがある SETPRINT マクロを生成する
/*JOBPARM	NOLOG		無視される
/*JOBPARM	PAGES		pages=*value* パラメータがある SETPRINT マクロを生成する
/*JOBPARM	PROCLIB		無視される
/*JOBPARM	RESTART		無視される
/*JOBPARM	ROOM		room=*value* パラメータがある SETPRINT マクロを生成する
/*JOBPARM	SYSAFF		無視される
/*JOBPARM	TIME		無視される

表 1-3　サポートされる JES2 文 (2 / 3)

文	パラメータ	使用法	トランスレータのアクション
/*OUTPUT		1 つ以上の sysout データセットに対する属性およびオプションを定義する	
/*OUTPUT	BURST		burst=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	CHARS		chars=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	CKPTLNS		ckptlns=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	CKPTPGS		ckptpgs=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	COMPACT		compact=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	COPIES		copies=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	COPYG		copyg=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	DEST		dest=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	FCB		fcb=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	FLASH		flash=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	FLASHC		flashc=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	FORMS		forms=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	INDEX		index=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	LINDEX		lindex=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	LINECT		linect=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	MODIFY		modify=*value* パラメータがある SETPRINT マクロを生成する

表 1-3　サポートされる JES2 文 (3 / 3)

文	パラメータ	使用法	トランスレータのアクション
/*OUTPUT	MODTRC		modtrc=*value* パラメータがある SETPRINT マクロを生成する
/*OUTPUT	UCS		ucs=*value* パラメータがある SETPRINT マクロを生成する
/*ROUTE		sysout データセットを経路指定する	SETPRINT dest=*xxxx* マクロを生成する

第2章

VSE JCL

この章の内容は、次のとおりです。

各文の構文についての説明は、IBM VSE マニュアルを参照してください。

変換例については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

VSE JCL 文のタイプ

VSE JCL 文には、次のようなタイプがあります。

ジョブエントリ制御言語 (JECL) 文	この文は、ジョブ入力、文の収集とスケジューリング、および出力スプール処理を管理するための VSE POWER JCL ストリームで使用される
ジョブ制御文 (JCS)	文の最初の 2 つの位置が `//` でコーディングされ、ジョブストリームにある
ジョブ制御コマンド (JCC)	文の最初の 2 つの位置に `//` なしにコーディングされ、ジョブストリームにありうる
アテンションルーチン (AR)	文の最初の 2 つの位置に `//` なしに `SYSLOG` (コンソール) から入力され、ジョブストリームにはない

JCL ストリームに入力できる文は、JCS または JCC、あるいはその両方です。オペランドの書式は通常同じですが、JCL で使用されるタイプにより実行時の動作が異なることがあります。

VSE JCL 文のサポート

VSE JCL 文の多くはメインフレーム固有であり、Sun MBM 環境には適用されません。これらの文はトランスレータによって変換されず、変換プロセス時に警告が発せられます。そのメッセージについては、『Sun Mainframe Batch Manager ソフトウェア メッセージガイド』を参照してください。

サポートされる VSE JCL 文

Sun MBM は、次の表に示す VSE JCL 文をサポートします。

表 2-1 サポートされる VSE JCL 文

文	オペランド	型	使用法	トランスレータのアクション
/.		JCC、JCS	GOTO または ON 文によって参照されるターゲットラベルを定義する	LABEL マクロを生成する
/+		JCS	プロシージャーの終了を指定する	ENDPROC マクロを生成する
/*		JCS	インストリームデータの終了を指定する	通常は感嘆符 (!) を生成する
/&		JCS	ジョブの終了を指定する	ENDJOB マクロを生成する
*		JCS	システムログにあるコメント行を指定する	LOGMSG マクロを生成する
ASSGN		JCS、JCC	論理ユニット (論理名) が物理入出力装置または別の論理ユニットに関連付けられるようにする	ASSGNDD マクロを生成する
DATE		JCS	VSE 環境で SET コマンドによって設定されたシステム日付を上書きする	SETDATE マクロを生成する
DLBL		JCS	プログラムで使用される論理名をディスクデータセットのラベル情報に対応付ける	DD マクロを生成する
EXEC		JCC、JCS	アプリケーションプログラム、システムユーティリティー、またはプロシージャーを実行する。ジョブステップの最後の文/コマンドであり、複数の物理行に渡って継続が可能	
EXEC	PGM			EXECPGM マクロを生成する
EXEC	PROC			EXECPROC マクロを生成する

表 2-1　サポートされる VSE JCL 文 (続き)

文	オペランド	型	使用法	トランスレータのアクション
GOTO		JCC、JCS	JCL 文をバイパスし、ジョブまたはプロシージャー内の指定されたラベルで実行を開始する	GOTO マクロを生成する
IF		JCC、JCS	ローカルな条件をチェックし、条件が真の場合に指定されたアクションを行う	IF/THEN マクロ構造を生成する
INCLUDE		JECL	名前の付けられたファイルからデータを変換時に組み込む	指定されたファイルの内容に INCLUDE 文を置換する
JOB		JCS	ジョブの開始を指定する。JOB 文がジョブストリームから省略された場合、ジョブ終了時にそのステップに対して期間や日付が出力されない	BEGINJOB マクロを生成する
LIBDEF		JCC、JCS	特定タイプのメンバーに対してサブライブラリチェーン (検索パス) を定義する	LIBDEF マクロを生成する
LIBDROP		JCC、JCS	1 つ以上の LIBDEF 文によって定義された検索パス情報を取り消す	LIBDROP マクロを生成する
LIBLIST		JCC、JCS	LIBDEF 文で定義された検索パス情報を表示する	LIBLIST マクロを生成する
LOG		JCC、JCS	JCS のログを開始する。JCC として入力すると、入力したパーティションの SYSLOG (コンソール) に後続のすべての JCL 文およびコマンドが書き込まれる	LOGMSG マクロを生成し、それによってジョブの履歴ファイルにコメントが書き込まれる
NOLOG		JCC、JCS	大部分の JCS のログを停止する	LOGMSG マクロを生成する
ON		JCC、JCS	特定条件との合致を検出した場合に行うアクションを指定する	条件に応じて、ONCONCODE または ONRETCODE マクロを生成する
PAUSE		JCC、JCS	現在のジョブの処理を一時的に中断する	PAUSE マクロを生成し、それによって msg パラメータのテキストがコンソールおよびジョブの履歴ファイルに送信される
PROC		JCC、JCS	プロシージャーの開始を指定し、そのプロシージャーによって使用されるパラメータのデフォルト値を提供する	BEGINPROC マクロを生成する
SET		JCC	プログラムの実行に関する処理制御を設定する	
SET	DATE			SETDATE マクロを生成する

表 2-1　サポートされる VSE JCL 文 (続き)

文	オペランド	型	使用法	トランスレータのアクション
SET	UPSI			SETPGMSW マクロを生成する
SET	LINEOT			無視される
SET	RF			無視される
SETPARM		JCC、JCS	記号パラメータに値を定義し、その値を割り当てる。この文では継続が可能	SETPARM マクロを生成する
TLBL		JCS	テープファイルのチェックおよび書き込みに関するファイルラベル情報を含める	ASSGNDD マクロを生成する
UPSI		JCS	後続のプログラムでテスト可能な 1 バイトセットのスイッチを設定する	SETPGMSW マクロを生成する

サポートされない VSE JCL 文

VSE JCL 制御オペレーションおよびコマンドの多くは、物理ハードウェアに関連し、Sun MBM 環境には適用されないメインフレーム固有の要件があります。次に例を示します。

- アプリケーションファイルによって使用される領域、またはプログラム実行の領域の管理 (EXTENT、SIZE、VDISK 制御オペレーション)
- システム領域の管理 (ALLOC、ALLOCR、MAP、QUERY 制御オペレーション)
- 基本ハードウェア制御 (SETPRT オペレーション)
- オペレーティングシステム機能 (SYSDEF、PRTY、START オペレーション)

アテンションルーチン (AR) は、Sun MBM 環境では適用されません。ただし、必要な場合には同等の機能を使用できます。

次の表は、サポートされない VSE JCL 文のリストの一部です。

表 2-2 サポートされない VSE JCL 文

文	型	使用法	理由
ALLOC	JCC	記憶域を静的パーティションに割り当てる	Sun MBM 環境では適用されないため
CANCEL	JCC	コマンドが発行されたパーティションで実行している現在のジョブを取り消す	Sun MBM 環境でジョブを取り消すには、Sun MBM コマンド `abtjob` を使用するため
CLOSE	JCC、JCS	ディスク、フロッピーディスク、またはテープに割り当てられた論理ユニットを閉じる	Sun MBM 環境では適用されないため
DVCDN	JCC	デバイスをシステムで使用不可にする。既存のすべての割り当てを削除する	Sun MBM 環境では適用されないため
DVCUP	JCC	割り当てに対してデバイスを使用可能にする	Sun MBM 環境では適用されないため
EXTENT	JCC、JCS	ディスクファイルの領域を定義する (ディスクファイル領域管理)	Sun MBM 環境では適用されないため
HOLD	JCC	フォアグラウンドパーティションをアンバッチするコマンドを発行するまで、割り当てまたはサブライブラリ定義を保持する	Sun MBM 環境では適用されないため
ID	JCC、JCS	ユーザー ID およびパスワード。どちらかが無効の場合、ジョブを取り消す。特定の保護されたリソースにアクセスするジョブに対して、この文は必須	Sun MBM では、セキュリティーがリソースレベルでチェックされ (ディレクトリまたはファイルに対するアクセス権)、バッチジョブ実行前に設定されるため。 Sun MBM は、指定されたユーザー ID だけに特定の Sun MBM コマンドの発行を限定することにより、セキュリティーの付加レベルを実現する
IGNORE	JCC	条件を無視するコマンドの入力をオペレータに許可する	Sun MBM `display` および `rpljob` コマンドによって機能が置き換わるため
JCLEXIT	JCC	ジョブでの各 JCC または文の読み取り時に開始する出口ルーチンを決定する	サポートされていないため
LISTIO	JCC、JCS	論理ユニットおよび物理ユニットの状態を一覧表示する、すべての I/O 割り当てのリストを作成する	Sun MBM 環境では適用されないため
MAP	JCC	さまざまな記憶域のマップを作成する	Sun MBM 環境では適用されないため
MSECS	JCC	パーティションのタイムスライスを表示または修正する	Sun MBM 環境では適用されないため
MTC	JCC、JCS	テープオペレーションを制御する	Sun MBM では、すべてのテープデータセットはディスクファイルとして処理されるため

表 2-2　サポートされない VSE JCL 文 (続き)

文	型	使用法	理由
NPGR	JCC	静的パーティションに割り当てられるプログラマ論理ユニット数	Sun MBM 環境では適用されないため
OPTION	JCS	システムのデフォルトオプションを一時的に上書きする	現在の Sun MBM リリースではサポートされていないため
OVEND	JCC、JCS	プロシージャー上書き文を区切る	現在の Sun MBM リリースではサポートされていないため
PRTY	JCC	パーティションおよびクラスに対するジョブ実行の優先順位を設定する。VSE では、この文はシステム初期設定時にだけ実行可能。	Sun MBM では、subjob または unikixjob Sun MBM コマンドの -p *priority* オプションおよび -c *class* オプションを使用して優先順位が割り当てられるのは、クラスに対してではなくジョブに対してであるため。指定されたクラスで複数のジョブが実行可能な場合、割り当てられた優先順位によってクラス内の処理順序が決定される
PWR	JCC、JCS	POWER コマンドをジョブストリームに渡す	Sun MBM 環境では適用されないため
QUERY	JCC、JCS	データ領域情報を表示する	Sun MBM 環境では適用されないため
RESET	JCC、JCS	RESET が発行されたパーティションに対して、一時的なサブライブラリ定義および I/O 割り当てを固定値にリセットする	現在の Sun MBM リリースではサポートされていないため
ROD	JCC	遠隔通信以外の装置に対して統計データのレコードカウンタを記録する	Sun MBM 環境では適用されないため
RSTRT		特定のチェックポイントからプログラムを再起動する	Sun MBM には、ジョブの再起動に代替方法があるため。subjob/unikixjob コマンドの -r オプションを使用して特定のステップまたはラベル名で再起動する
SETPFIX	JCC、JCS	PFIXing ページ制限を 16M バイトを基準に定義する	Sun MBM 環境では適用されないため
SETPRT	JCC、JCS	IBM 3800 印刷サブシステムの制御値を設定する	現在の Sun MBM リリースではサポートされていないため
SIZE	JCC	パーティションでのプログラム実行のために予約される連続仮想記憶域の量	Sun MBM 環境では適用されないため
START	JCC	パーティションを開始する	Sun MBM では、ジョブはパーティションではなくクラスで実行されるため。並行ジョブを実行するためのクラスを設定可能
STDOPT	JCC、JCS	システム初期設定で設定された固定ジョブ制御オプションを設定またはリセットする	Sun MBM 環境では適用されないため

表 2-2　サポートされない VSE JCL 文 (続き)

文	型	使用法	理由
STOP	JCC	コマンドが発行されたパーティションでジョブの実行を中断する	Sun MBM では、ジョブはアクティビティーで実行されるため。BAM を使用することにより、クラスでジョブの実行を中断できる
SYSDEF	JCC、JCS	制限およびデータスペースを定義する	Sun MBM 環境では適用されないため
UCS	JCC	汎用文字セットバッファーを読み込む	Sun MBM 環境では適用されないため
UNBATCH	JCC	フォアグラウンド処理を終了し、パーティションを解放する	Sun MBM では部分的にサポートされる。クラスの作成または削除は、BAM を使用して可能。UNBATCH と関連付けられた論理ユニット管理は、Sun MBM ではサポートされない
VDISK	JCC、JCS	仮想ディスクの配置を定義し、初期化する (ディスクファイル管理)	Sun MBM 環境では適用されないため
ZONE	JCS	地方時とグリニッジ標準時との時差を指定する	サポートされていないため

注 – CLOSE および DVCDN は、固定論理ユニット割り当てに影響しますが、現在サポートされていません。

ジョブエントリ制御言語文

VSE/POWER ジョブのジョブエントリ制御言語 (JECL) 文は、ジョブ入力文の収集およびスケジューリングを可能にし、プログラム実行からの出力のスプール処理を制御します。この文は、接頭辞 '* $$' で始まります。使用可能な JECL 文の一部が Sun MBM によってサポートされます。

次の表に示す文は、現在のバージョンの Sun MBM でサポートされます。

表 2-3 サポートされる JECL 文

文	使用法	トランスレータのアクション
* $$ DATA	* $$ SLI 要求によって取り出されるライブラリメンバーにデータを挿入する	データファイルを挿入する
* $$ SLI	JCL を含む SLI メンバーを * $$ SLI 文に代わって現在のジョブストリームに組み込めるようにする	JCL ファイルを挿入する
* $$ LST	sysout ファイルにプリンタ属性を割り当て、適用できるようにする	SETPRINT マクロを生成する

次の表に示す文は、現在のバージョンの Sun MBM でサポートされず、dostrans によって無視されます。

表 2-4 サポートされない JECL 文

文	説明
* $$ CTL	実行のためにデフォルトクラスを割り当てる
* $$ EOJ	VSE/POWER ジョブの終了を指定する
* $$ FLS	VSE/POWER ジョブを即時終了する
* $$ JOB	VSE/POWER ジョブの開始を指定する
* $$ PUN	パンチ/出力属性を定義する
* $$ RDR	入力ジョブに 3540 ファイルを追加する

第3章

Sun MBM マクロ

Sun MBM には、バッチジョブを作成するために使用できるマクロが提供されています。これらのマクロは、実行するアプリケーションプログラムおよび必要なリソースを記述します。一連のマクロ文はファイルにまとめ、それをサブシステムにサブミットします。この章では、マクロ構文について説明します。表 3-1 は、マクロの概要を示しています。

ジョブエディタを使用して新しいマクロジョブを作成する方法については、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

表 3-1 Sun MBM ジョブマクロセット

マクロ	説明
ASSGNDD	アプリケーションプログラムがファイルへのアクセスを必要とする場合、外部名とファイルを関連付ける
BEGINJOB	ジョブの開始を定義する必須エントリ
BEGINPROC	プロシージャーの開始を定義する、プロシージャーの必須エントリ
DISPLAY	定義済み Sun MBM コンソールおよび Sun MBM ジョブの履歴ファイルにメッセージを表示する
EBMSYSCMD	マクロジョブスクリプトからシステムコマンドを実行する
ENDJOB	ジョブの終了を定義する必須エントリ
ENDPROC	プロシージャーの終了を定義する、プロシージャーの必須エントリ
EXECPGM	アプリケーションプログラムまたはユーティリティーを実行する
EXECPROC	プロシージャーを実行する
GOTO	後続ステップをバイパスし、指定された名前で実行を再開する
IF/THEN IF/THEN/ELSE/ENDIF	条件、およびその条件が真の場合に実行するアクションを定義する
LABEL	ジョブまたはプロシージャーストリーム内で、実行を再開または中断する特定ポイントを定義する

表 3-1　Sun MBM ジョブマクロセット (続き)

マクロ	説明
LASTRC	実行された最後のステップのリターンコードを取り出す。IF/THEN マクロとともに使用する
LIBDEF	実行可能プログラムまたはプロシージャーの検索パスを定義する
LIBDROP	プロシージャーまたは実行可能プログラムへの検索パスをデフォルトディレクトリにリセットする
LIBLIST	プロシージャーまたは実行可能プログラムに対する現在の検索チェーンを表示する
LOGMSG	Sun MBM ジョブの履歴ファイルにメッセージを書き込む
MAXRC	ジョブで前に実行された全ステップのうちの最大リターンコードを返す。IF/THEN マクロとともに使用する
ONCONDCODE	前のステップのどれかによって発行された条件コードに基づいて実行するアクションを定義する
ONRETCODE	前のステップによって発行されたリターンコードに基づいて実行するアクションを定義する
PAUSE	ジョブの実行を中断する
SETDATE	ジョブの期間に関する Sun MBM システム日付を上書きする
SETPARM	記号パラメータに値を割り当てる
SETPGMSW	Micro Focus COBOL のプログラム可能なスイッチ COBSW に値を割り当てる
SETPRINT	システム出力ファイルの生成時に参照される一連の印刷オプションを定義する
SETRETCODE	現在のステップのリターンコードをユーザー作成ユーティリティーから使用された数値に設定する

マクロによるジョブおよびプロシージャーの作成

この節では、Sun MBM マクロ文を使用してジョブおよびプロシージャーを作成する方法の概要について説明します。

ジョブの作成

Sun MBM ジョブは、一連のマクロ文を含むファイルです。ish ディレクトリに作成する各ジョブには、次のマクロ文を含める必要があります。

```
BEGINJOB
ENDJOB
```

これにより、subjob または unikixjob コマンドを使用してそのファイルをサブシステムにサブミットできます。このコマンドについては、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

バッチジョブには通常、一連のステップを記述します。各ステップでは、ファイル割り当てを指定し、そのステップで実行するアプリケーションプログラムおよびユーティリティーを定義します。Sun MBM 環境では、ステップに割り当てる必要があるすべてのファイルに対して ASSGNDD マクロ文を使用し、ステップで実行するプログラムまたはユーティリティーを定義するために EXECPGM マクロを使用します。

たとえば次のように、1 つのジョブに 2 つのアプリケーションプログラム CB001 および CB002 を定義して実行します。最初のプログラム CB001 には、2 つのファイル VSINP1 および VSOUT1 が必要です。2 番目のプログラム CB002 には、1 つのファイル VS102 が必要です。また、ジョブの開始と終了のために、BEGINJOB および ENDJOB マクロを使用します。

```
BEGINJOB
ASSGNDD ddname=INPUT1 filename=VSINP1 type=VS
ASSGNDD ddname=OUTPUT1 filename=VSOUT1 disp=o type=VS
EXECPGM pgmname=CB001 stepname=STEP001
ASSGNDD ddname=INPOUT2 filename=VS102 type=VS
EXECPGM pgmname=CB002 stepname=STEP002
ENDJOB
```

注 – マクロ文の順番は、DD 文より先に EXEC 文を置く MVS JCL とは逆になります。

ジョブがサブミットされると、Sun MBM バッチシェルコマンド btsh によって解釈されます。したがって、display、accept などの btsh 組み込み型文 (このコマンドについては、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照) や、有効な C シェル文およびシステムコマンドをジョブに含めることができます。これらのコマンドをユーザー定義のユーティリティーにカプセル化して EXECPGM マクロで実行したり、EBMSYSCMD マクロを使用したりできます。

自動ジョブ中断、システム出力ファイル (SYSOUT)、アプリケーションプログラムおよびプロシージャーの代替検索パス、データ割り当てなど、マクロによってサポートされる追加機能もマクロセクションに記述できます。

プロシージャーの作成

ステップの共通部分を 1 つのプロシージャーにまとめることにより、それを別々のジョブで実行できます。Sun MBM プロシージャーには通常、一連のマクロを含めます。また、Sun MBM バッチシェル組み込み型コマンド、または有効な C シェル文およびシステムコマンドも指定できます。これらのコマンドをユーザー定義のユーティリティーにカプセル化して EXECPGM マクロで実行したり、EBMSYSCMD マクロを使用したりできます。

プロシージャーのファイルは、PROCLIB 環境変数で定義されたディレクトリに作成します。そのファイルには、次の 2 つのマクロ文を含める必要があります。

```
BEGINPROC
ENDPROC
```

これにより、EXECPROC マクロ文を使用してプロシージャーを呼び出せます。

例

次の例では、$PROCLIB ディレクトリのファイルには PROC1 という名前が付いています。次のように文を指定することにより、どのジョブでもそのファイルを実行できます。

```
BEGINJOB
...
(必要に応じてマクロ文を追加)
...
EXECPROC procname=PROC1 parms='A=1,B=2'
...
ENDJOB
```

次の例では、CB003 という名前のプログラムを実行するプロシージャーを定義しています。また、プロシージャーの開始と終了のために、BEGINPROC および ENDPROC マクロを使用します。デフォルトの記号パラメータ値は BEGINPROC マクロで定義されています。この例では、そのパラメータの値は EXECPROC マクロで定義された値によって上書きされます。

```
EGINPROC  parms='A=A,B=B'
ASSGNDD   ddname=INP1 filename=/date/PRIMDT
EXECPGM   pgmname=CB003
ENDPROC
```

ステップおよびジョブ実行の制御

デフォルトでは、1 つのステップが異常終了した場合にすべてのマクロ文をバイパスすることによって、Sun MBM はジョブ実行を制御します。ONCONDCODE および ONRETCODE マクロを使用することにより、ステップの順番を CONDITION-CODE および RETURN-CODE レベルで制御することもできます。IF/THEN、GOTO などのその他のマクロでは、ステップのリターンコードまたは異常終了した前のステップに基づいて、ステップ実行を柔軟に制御できます。

ジョブ実行を制御するマクロ文は、要件に応じて指定できます。Sun MBM マクロではないジョブの文のすべては、この条件論理に依存しません。したがって、EBMSYSCMD マクロを使用するか、これらの文をユーザーユーティリティーにカプセル化して EXECPGM マクロで実行します。ユーザーユーティリティーの詳細については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

ジョブでの実行を再開または中断するポイントを LABEL マクロを使用して定義します。

スクリプトの妥当性のテスト

ジョブの作成後、ジョブおよびすべての関連プロシージャーを検査モードでサブミットすることによって、ジョブの妥当性を検査できます。これにより、使用可能な段階ではなくても、すべてのアプリケーションプログラムの実行およびすべての必要なファイルの割り当てを Sun MBM によってシミュレーションできます。検査モードでのジョブのサブミットについては、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

マクロ文の継続

各マクロ文は 1 行でコーディングするか、継続文字 \\\ に次の行を続けて複数行に渡ってコーディングします。

次に例を示します。

```
BEGINJOB
ASSGNDD  ddname=OUTFILE filename=/test/outfile
ASSGNDD  ddname=INFILE                    \\\
         type=FS                          \\\
         filename=/test/projecta/data     \\\
         disp=i                           \\\
         abend=k normal=k
```

注 – パラメータとその値は、1 つのオペランドとして扱われます。継続文字によってオペランドを中断することはできません。

特殊文字を使用する場合は、付録 A を参照してください。

ASSGNDD

ASSGNDD マクロは、外部名をファイルに関連付けます。アプリケーションプログラムが必要なファイルをジョブによって外部から割り当てることを求める場合、このマクロを使用します。

形式

```
ASSGNDD ddname=external-file-name
alias=alias-external-file-name [verbose]
```

または

```
ASSGNDD [ddname=external-file-name|ddname='procstepname.external-file-name']
[abend=action]
[class=class-id|JOBCLASS]
[copies=number]
[dataset=dataset-name]
[[dest=dest-name]|[dest='node,userid']]
[disp=file-access]
[filename=file-name[:file-name:...]
[gdg=gdg-number|ALL]
[member=member-name]
[normal=action]
[printid=setprint-identifier]
[recfmt=F|V]
[recsize=record-size]
[rls=CR|NRI]
[type=FS|VS|INSTREAM|SYSOUT|DUMMY|UNASSIGN|TEMP]
[verbose]
```

または

```
ASSGNDD ddname=external-file-name type=INSTREAM << !
instream-data
...
!
```

説明

ddname=*external-file-name*|ddname=*'procstepname.external-file-name'*

external-file-name

1 ～ 8 文字の英数字のファイル名であり、次のようにアプリケーションプログラムで定義します。次に例を示します。

```
ASSGNDD ddname=INVMTD...
```

COBOL では、この名前は、次のように SELECT 文の ASSIGN TO オプションの後ろに指定される名前に対応します。次に例を示します。

```
SELECT INFILE ASSIGN EXTERNAL INVMTD
 ORGANIZATION IS RECORD SEQUENTIAL.
```

PL/I では、この名前は、次のように OPEN 文の TITLE オプションに続く名前に対応します。次に例を示します。

```
OPEN FILE(INFILE) RECORD KEYED SEQUENTIAL
 OUTPUT TITLE('INVMTD');
```

DD の連結

データセットを連結するには、ddname を使用して連結の最初のファイルを指定することによって ASSGNDD をコーディングします。後続の各ファイルについては、ddname を指定しないで、datasetname または filename を入力することによって ASSGNDD マクロをコーディングします。ファイルが連結の一部ではない場合は、ddname は必須です。サポートされているのは COBOL プログラムのみです。

'procstepname.external-file-name'

DD の上書き

プロシージャー ddname を上書きするには、ジョブレベルの ddname を次のようにコーディングします。

ddname=*proc-stepname.dataset-name*

次の例では、ジョブレベルの ASSGNDD オプションが、INFILE に対するプロシージャーレベル ASSGNDD のオプションを上書きします。

ジョブレベル

```
BEGINJOB ...
...
LABEL name=JOBSTEP1
...
ASSGNDD ddname=PROCSTP1.INFILE ...
...
```

プロシージャーレベル

```
BEGINPROC ...
...
LABEL name=PROCSTP1
...
ASSGNDD ddname=INFILE ...
```

DD 上書きは、SETPRINT マクロで指定される SYSOUT 印刷オプションでも使用できます。これについての説明は、69 ページの「SETPRINT」マクロを参照してください。

abend=*action*

ステップが異常終了した場合に行うアクションです。

d: ファイルを削除する

k: ファイルを保持する。デフォルト

alias=*alias-name*

現在の ddname を *alias-name* に関連付けられたファイルに割り当てます。この *alias-name* は、前の ASSGNDD マクロ文で割り当てられた ddname です。

ddname および alias 以外のすべてのオペランドは無視されます。*alias-name* に関連付けられた ddname が現在割り当てられている場合、チェックは行われません。

class=*class-id*|JOBCLASS

type=SYSOUT データセットを特定の出力クラスに割り当てます。JOBCLASS が定義されている場合、BEGINJOB マクロの class= オプションによって指定されるクラスにそのデータセットが割り当てられます。

copies=*copies*

印刷する部数であり、type=SYSOUT データセットの場合にだけ有効です。copies= オプションは、SETPRINT マクロを使用しても定義できます。デフォルトは 1 です。上書きルールについては、73 ページの「印刷出力の管理」を参照してください。

dataset=*mainframe-dataset-name*

メインフレーム上のデータセットのオリジナル名です。次に例を示します。

PROD.INVENTORY.MASTER

オリジナルの**メインフレーム上のデータセット名**を参照してファイルにアクセスするメインフレームジョブ (IDCAMS REPRO、DELETE、DEFINE ユーティリティーなど) を移行する場合に、このオプションパラメータを使用します。

詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』の IDCAMS および SORT ユーティリティーの説明を参照してください。

type=FS または VS の場合、dataset または filename、あるいはその両方を ASSGNDD で指定する必要があります。filename を指定しない場合は、dataset によって実行されるときに Sun MBM File_Map から取り出されます。

dest=*dest-name*|dest='*node,userid*'

出力ファイルの印刷先を定義するプリンタ名であり、type=SYSOUT データセットの場合にだけ有効です。dest オプションは、SETPRINT マクロを使用して定義できます。上書きルールについては、73 ページの「印刷出力の管理」を参照してください。

disp=*file-access*

ファイルアクセスには、次の方法があります。

i: 入力用にファイルを開く。デフォルト

o: 出力用にファイルを開くデフォルトでは、出力ファイルが存在する場合、それは削除され、ジョブが継続されます。既存の出力ファイルを保持し、ジョブを強制的に中止させるには、ジョブ設定ファイルまたはサブシステムのユーザー設定ファイルの ABEND に、変数 OUTPUT_FILE_EXISTS を設定します。

i-o: 入力および出力用にファイルを開く

a: 追加モードでファイルを開く

COBOL でファイルを追加書き込みする場合は、OPEN 文の EXTEND オプションを指定する必要があります。

filename=*system-file-name*

dataset を指定しない場合は、type=FS および VS に対して filename= が必要です。

type=SYSOUT では無視されます。

type=DUMMY ではオプションです。

type=FS の場合、システムファイルのフルパス名を次のように指定します。次に例を示します。

```
filename=/prod/inv/mtd/summary
```

type=VS の場合、Sun MTP ファイル管理テーブル (FCT) で定義されているのと同じ VSAM ファイル名を次のように指定します。次に例を示します。

```
filename=CUSTFILE
```

gdgnum=*'gdg-number'*|ALL

COBOL でアクセスされる type=FS ファイルの場合だけ有効です。デフォルト

type=FS であり、ファイルが File_Map で GDG として定義されている場合、*gdg-number* は、アクセスするファイルの特定オカレンスを表します。*gdg-number* は、0、+*n*、または -*n* (*n* は 1 ～ 128 までの整数) です。

そのファイルのグループに対する GDG オカレンスの最大数を指定するために、File_Map を更新する必要があります。手順については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

ファイルが GDG として File_Map で定義され、gdgnum=*option* が指定されていない場合、または gdgnum='ALL' として指定されている場合、すべての GDG オカレンスは 1 つの入力ファイルとして連結されます。

member=*member-name*

dataset によって指定されるデータセットのメンバー名です。dataset が指定されている場合だけ有効です。File_Map のエントリは、dataset(*member-name*) で表示されます。次に例を示します。

TEST.DATA(MEMBER1)

normal=*action*

ステップが正常に終了した場合に行うアクションです。

d: ファイルを削除する

k: ファイルを保持する。デフォルト

printid=*setprint-identifier*

前の SETPRINT マクロへの明示的な参照です。SYSOUT ファイルは、ASSGNDD マクロが暗黙的または明示的に参照する SETPRINT マクロによって定義された印刷属性を継承します。type=SYSOUT に対してだけ有効です。ASSGNDD マクロによる SETPRINT マクロの参照については、73 ページの「印刷出力の管理」を参照してください。

recfmt=F|V

レコードの次の書式を指定します。

F: 固定長。デフォルト

V: 可変長

SORT または IDCAMS ステップでのファイル割り当てに対してだけ必要です。

さらに、Sun MBM は *ddname*_RECFMT 変数を次のどちらかに設定します。

record: 固定長レコード

recordv: 可変長レコード

recsize=*record-size*

固定長ファイルのレコード長です。可変長ファイルの場合は、最大レコード長を表します。SORT または IDCAMS ユーティリティーによって使用されるファイルの割り当てで必要になる場合があります。これらのユーティリティーおよびその要件については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

さらに、Sun MBM は *ddname*_LRECL 変数を定義済み *record-size* に設定します。

rls=CR|NRI

レコードレベルシェアリングのオプションです。サブシステムで RLS オプションセットを設定することにより、このオプションに関するメインフレーム互換性を有効にする必要があります。BAM を使用して RLS を設定する方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

CR:

整合性を保った読み取り (Consistent Reads) を指定。アプリケーションが読み取る各レコードで、VSAM がロックを取得する

NRI:

整合性に留意しない読み取り (No Read Integrity) を指定。アプリケーションは全レコードの読み取りが可能

type=

次のファイルタイプを指定します。指定しない場合のデフォルト値は FS です。

FS:

システムファイル。デフォルト

VS:

VSAM ファイル

INSTREAM:

ASSGNDD マクロに渡されるインストリームデータとしてデータセットが定義される

SYSOUT:

システム出力ファイルを作成および印刷する

DUMMY:

filename= パラメータを上書きする。入力ファイルの場合、最初の読み取りアクセスで EOF が返され、出力ファイルが作成されない

TEMP:

一時ファイルをシステムの一時ディレクトリに作成する。type=UNASSIGN の後続 ASSGNDD が検出されるまで、またはジョブ終了までファイルは保持される

デフォルトでは、データセットに関して File_Map にエントリは生成されない。ジョブが終了すると、ファイルは常に削除される

dataset および filename はオプション。ASSGNDD マクロで filename を指定すると、その名前が使用される。指定しない場合は、Sun MBM が一意のファイル名を生成する

一時ファイルは、環境変数 EBMTMPDIR によって定義されたディレクトリに作成される。デフォルトでは、$EBMTMPDIR はプラットフォームのデフォルト一時領域に設定される。Sun MBM 管理者は、BAM を使用してサブシステムを更新することによってこの設定の変更が可能。詳細については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照

UNASSIGN:

ddname の現在の割り当てを設定解除する

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

注 – BEGINJOB マクロで mode=MVS を指定すると、ステップ終了時にすべてのファイル割り当てが解除されます。指定しない場合は、ジョブ終了まで、または type=UNASSIGN の ASSGNDD が実行されるまでファイル割り当ては有効です。

例

入力ファイルの割り当て:

```
ASSGNDD ddname=INFILE type=FS  \\\
        filename=/test/customer/master
```

GDG オカレンスによる出力ファイルの割り当て:

```
ASSGNDD ddname=OUTFILE type=FS gdgnum='+1'          \\\
        filename=/test/customer/mtd disp=o abend=d  \\\
        normal=k
```

入力ファイルとして GDG の全オカレンスの割り当て:

```
ASSGNDD ddname=INFILE gdg=ALL dataset=TEST.INV  \\\
```

SYSOUT ファイルの割り当て:

```
ASSGNDD ddname=MTDRPRT type=SYSOUT dest=RMT1  \\\
        copies=2
```

ダミーファイルの割り当て:

```
ASSGNDD ddname=INV0003 type=DUMMY  \\\
```

現在の割り当てを設定解除するために UNASSIGN タイプを使用:

```
ASSGNDD ddname=SYS0003 type=UNASSIGN
```

入力および出力用の Sun MTP VSAM ファイルの割り当て:

```
ASSGNDD ddname=INFILE type=VS filename=CUSTFILE \\\
        disp=i-o
```

インストリームデータの割り当て:

```
ASSGNDD ddname=SYSIN type=INSTREAM \\\
        disp=i abend=d normal=d <<!
AX WEST 200
MX MW 400
!
```

連結の入力ファイルの割り当て:

```
ASSGNDD ddname=INFILE type=FS filename=/test/cust \\\
ASSGNDD filename=/test/cust2                      \\\
ASSGNDD filename=/test/cust3
```

エイリアスの割り当て:

```
ASSGNDD ddname=SYS003 type=FS disp=i abend=d \\\
        normal=k
...
ASSGNDD ddname=SYS005 alias=SYS003
```

BEGINJOB

このマクロは、ジョブの開始を定義します。必須エントリです。

形式

```
BEGINJOB jobclass=class
programmer=jobowner
[mode=VSE|MVS]
[username=userid] [password=password]
[verbose]
```

説明

`jobclass=`

ジョブによって作成されるすべての SYSOUT ファイルに対して出力クラスを定義します。デフォルト値および上書きについては、73 ページの「印刷出力の管理」を参照してください。

`mode=VSE|MVS`

ジョブの次の動作を指定します。

VSE: VSE JCL の動作をシミュレーションする。デフォルト

MVS: MVS JCL の動作をシミュレーションする

一定のマクロがこのオプションに基づいてそれぞれに動作します。たとえば、VSE を指定すると、ジョブの期間中、または ASSGNDD マクロによって再割り当てや割り当て解除が行われるまで 1 つの割り当てが設定されます。

MVS を指定した場合は、ファイル割り当てはそれが定義されているステップの実行中のみ有効です。

メインフレームから移行する場合、モードは一定の設定ファイルへのアクセスにも影響します。

`password=`

ジョブをサブミットしたユーザー ID のパスワードを指定します。

programmer=

PROGRAMMER_NAME 環境変数に、ジョブをサブミットするユーザーではなくジョブの作成者を指定します。

username=

ジョブをサブミットしたユーザーを指定します。

verbose

マクロを冗長モードで実行します。環境変数の値およびその他のデバッグメッセージが、ジョブの履歴ファイルに書き込まれます。

例

オプションパラメータを使用してジョブを開始します。

```
BEGINJOB jobclass=A
```

この例では、ジョブによって生成される SYSOUT ファイルがプリンタクラス A に割り当てられます。

BEGINPROC

BEGINPROC マクロは、プロシージャーの開始を定義します。必須エントリです。

形式

BEGINPROC [procname=*procedure-name*]
[parms='*parmname=string*[,*parmname=string*...]']
[verbose]

説明

procname=

プロシージャーの名前です。

parms=

プロシージャーで参照される記号パラメータに初期値を割り当てます。

parmname: 1 ～ 7 文字の英数字の記号パラメータ名です。

string: 100 文字までの文字列とする記号パラメータの初期値です。パラメータ値を無効にするには、parmname= または parmname=" と入力します。

値を上書きするには、EXECPROC マクロを使用します。

次の書式によって、BEGINPROC マクロに複数の parms= エントリを指定できます。

```
EXECPROC procname= procname                          \\\
         parms=' parmname= value, parmname= value'  \\\
         par ms='parmname=value ,parmname=value'
```

verbose

マクロを冗長モードで実行します。環境変数の値およびその他のデバッグメッセージが、ジョブの履歴ファイルに書き込まれます。

例

オプションパラメータによるプロシージャーの開始:

```
BEGINPROC parms='SYS=prod,RUN=MTD'
```

parms= パラメータの連結:

```
BEGINPROC parms='SYS=prod,RUN=MTD'  \\\
          parms='ADF=YES,MO=MAY'
```

DISPLAY

DISPLAY マクロは、引用符付きテキストを Sun MBM ジョブ履歴ファイルおよびオプションで定義済み Sun MBM コンソールに書き込みます。

形式

DISPLAY [console=YES] '*comment*'

説明

console=YES

コメントを定義済みコンソールに書き込みます。

comment

ジョブの履歴ファイルおよびオプションで定義済み Sun MBM コンソールに書き込むテキスト。

注 – DISPLAY マクロで特殊文字を使用した場合の結果は予測できません。このマクロ文を生成する場合、VSE JCL トランスレータ dostrans はすべての特殊文字を下線に変換します。

LOG 文の詳細は、56 ページの「LOGMSG」および『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

EBMSYSCMD

EBMSYSCMD マクロは、Sun MBM RETURN-CODE および CONDITION-CODE 処理ロジックの制御のもとでマクロジョブスクリプト内からのシステムコマンドの実行を可能にします。EBMSYSCMD は、ONRETCODE または ONCONDCODE 条件に基づいて指定されたコマンドを実行します。

system command は、SHELL 環境変数で指定されるデフォルトのシェル、または EBMSYSCMDSH 環境変数で定義されるパスによって指定されるシェルスクリプト ($USER_SETUP, $PACK/btshrc または $HOME/.btshrc ファイルで指定) を使用して実行されます。Sun MBM で使用される設定ファイルの詳細については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

system command が複数行のエントリの場合、すべてのコマンド行が終了するかゼロ以外の終了状態がシェルから返されるまで、順次エントリが実行されます。ゼロ以外の終了状態が返されると、Sun MBM は 255 の強制中止コードを設定し、残りのステップをバイパスします。シェル状態に対してさらに制御が必要な場合は、EXECPGM マクロコマンドによって起動されるユーザーユーティリティーを使用します。

形式

```
EBMSYSCMD << !
system command
[system command]
!
```

説明

system command

マクロで実行する有効なコマンド。

注 – どのような場合に &、'、*、$ などの特殊文字をバックスラッシュでエスケープするかを、使用する特定のコマンドに応じてきめる必要があります。

例

Sun MBM ジョブ時のバックグラウンドシェルスクリプトの開始:

```
EBMSYSCMD << !
cp $DEVNULL $HOME/myfile
$HOME/myjob \&
!
```

ENDJOB

ENDJOB マクロは、ジョブを終了するために使用します。このマクロの機能は、次のとおりです。

- ジョブの終了を定義する
- ジョブの開始および完了時刻を表示する
- ジョブが正常に終了したか不正終了したかを示すジョブ状態を表示する
- ジョブの一時作業ファイルを削除する

形式

```
ENDJOB [verbose]
```

説明

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

ENDPROC

ENDPROC マクロは、プロシージャーを終了するために使用します。このマクロの機能は、次のとおりです。

- プロシージャーの終了を定義する
- プロシージャーの一時作業ファイルを削除する
- プロシージャーの入れ子を制御するタスクを実行する

形式

```
ENDPROC [verbose]
```

説明

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

EXECPGM

アプリケーションプログラムまたはユーティリティーを実行します。デフォルトでは、EXECPGM は parm= パラメータで指定された環境変数を展開します。各サブシステムの $USER_SETUP ファイルまたは $PACK/btshrc ファイルで次の環境変数を設定することにより、この動作を変更できます。

```
setenv EXECPGM_EXPAND_PARM N
```

Sun MBM で使用される設定ファイルの詳細については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

形式

EXECPGM pgmname=*program-name* [parm='*value*']
[stepname=*stepname*] [verbose]

説明

pgmname=*program-name*

実行するプログラムまたはユーティリティーの名前を指定します。必須エントリです。

EXECPGM は、指定された *program-name* をアプリケーションプログラムではなくユーティリティーとして実行するかを判定するために、最初に設定ファイルを検索します。BEGINJOB マクロで mode=MVS を指定した場合、$PUBLIC/mvs.conf ファイルが検索されます。mode=VSE を指定した場合、またはモードを指定しなかった場合、$PUBLIC/dos.conf ファイルが検索されます。

スクリプト名を適切な設定ファイルに追加してそのスクリプトを PATH 変数に含まれるディレクトリに配置すると、スクリプトをユーティリティーとして実行できます。サブシステムユーザー設定ファイルの PATH 変数を更新する必要があります。実行時にその名前が設定ファイルで見つからない場合は、アプリケーションプログラムとして実行されます。

parm=

アプリケーションプログラムに渡される 100 文字までのパラメータです。その文字列に含まれる特殊文字は、アポストロフィで囲む必要があります。アポストロフィはプログラムには渡されません。

stepname=

MVS モードで実行するジョブ:

ジョブステップの一意の識別子 (1 ～ 8 文字の英数字)。stepname が次の場合に必要です。

- GOTO マクロ文のターゲットである場合
- subjob および unikixjob コマンドの再起動または中断機能 (-R、-S オプション) を必要とする場合
- プロシージャーであり、DD 上書きを含むジョブによって実行できる場合。次に例を示します。

```
//PSTP001.DD1 DD DSN=JOB.Data ...
```

VSE モードで実行するジョブ:

GOTO マクロ文のターゲットとして stepname は使用できません。GOTO 文のターゲットは、LABEL マクロで定義する必要があります。

stepname を指定しなかった場合、*program-name* がデフォルトとして設定されます。

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

例

parm オプションによってデータをプログラムに渡す:

```
EXECPGM pgmname=ACCT0001 stepname=acctprnt parm='YTD'
```

EXECPROC

EXECPROC マクロは、プロシージャーを実行します。プロシージャーの作成方法については、36 ページの「BEGINPROC」を参照してください。

形式

```
EXECPROC procname=procedure-name
[parms='parmname=[value][,parmname=['value'],...]'
[pgmparm [ .procstepname]='value']]
[stepname=stepname]
[verbose]
```

説明

procname=*procedure-name*

実行するプロシージャーの名前です。必須エントリです。

parms='*parmname*=[*value*][,*parmname*=[*value*][,...]'

parmname: 呼び出されたプロシージャーに渡す記号パラメータ値の名前 (1 ～ 7 文字の英数字)。

value: 記号パラメータの値を表す文字列。

値を無効にするには、次のどちらかを入力します。

- parmname=" または parmname=
- 英数字以外の文字を含む *value* 文字列は、アポストロフィで囲む必要があります。

次の文字の前には、バックスラッシュ (\ ) を付ける必要があります。

* ( ) '

このオプションによって指定された値は、BEGINPROC マクロの parms オプションによって指定されたデフォルト値を上書きします。

次の書式によって、EXECPROC マクロに複数の parms= エントリを指定できます。

```
EXECPROC procname=procname                          \\\
         parms='parmname=value,parmname=value'     \\\
         parms='parmname=value,parmname=value'
```

pgmparm.*procstepname*=

呼び出されたプロシージャーのステップの EXECPGM マクロによって指定された parm データは、*value* によって上書きされます。

pgmparm=

呼び出されたプロシージャーの最初の EXECPGM マクロによって指定された parm データは、*value* によって上書きされます。他のすべてのプロシージャーステップの parm データは無効化されます。

stepname=

GOTO マクロ文によって参照される、あるいは subjob または unikixjob コマンドの引数として指定されたジョブまたはプロシージャースクリプト内の一意の名前。subjob および unikixjob コマンドの -R および -S オプションは、特定のステップにおいてジョブを再開または中断します。*stepname* を指定しなかった場合、*procedure-name* がデフォルトとして設定されます。

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

例

記号パラメータに値を割り当てるプロシージャーを実行します。この値は、BEGINPROC 文の parms オプションによって定義されたデフォルト値を上書きします。

```
EXECPROC procname=copycust stepname=cust0001 \\\
         parms='SYS=prod,DIR=/prod/cpy'
```

parms= オプションの連結:

```
EXECPROC procname=PROC2              \\\
         parms='SYS=prod,RUN=MTD'    \\\
         parms='ADF=YES,MO=MAY'
```

GOTO

GOTO マクロは、後続ステップをバイパスし、ジョブまたはプロシージャーの特定のポイントで実行を再開します。

形式

GOTO {END_JOB|*label-name*|*stepname*} [verbose]

説明

END_JOB

ジョブを終了します。制御が ENDJOB マクロに移ります。

label-name

実行を再開するジョブまたはプロシージャーのポイントを定義します。LABEL マクロで指定された名前である必要があります。

stepname (MVS JCL)

実行を再開するジョブまたはプロシージャーのポイントを定義します。EXECPGM または EXECPROC マクロの *stepname* パラメータで指定された名前である必要があります。

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

ジョブ内に GOTO を指定する場合、そのジョブ内に *stepname* または *label-name* を定義する必要があります。

プロシージャー内に指定する場合、同じプロシージャー内に *stepname* または *label-name* を定義する必要があります。

ジョブストリームでは、ラベル名またはステップ名を前方検索するだけです。*label-name* または *stepname* が見つからないと、ジョブは終了します。

例

LABEL および EXECPGM マクロによる GOTO の使用:

```
BEGINJOB
ASSGNDD  ddname=INDET type=FS   \\\
         filename=/prod/inventory/detail
ASSGNDD  ddname=INSUM type=FS    \\\
         filename=/prod/inventory/summary
ASSGNDD  ddname=INVREPT type=SYSOUT
EXECPGM  pgmname=INV0001 stepname=INV0001
EXECPGM  pgmname=INV0002 stepname=INV0002
IF MAXRC GT 4
      THEN GOTO ABENDJOB
EXECPGM  pgmname=INV0003 stepname=UPDINV
IF LASTRC EQ 0
      THEN GOTO GENRPT
GOTO AB9999
LABEL ABENDJOB
EXECPGM  pgmname=ABHANDLE stepname=AB9999
GOTO     END_JOB
LABEL GENRPT
EXECPGM  pgmname=RPT0001 stepname=INVREPT
EXECPGM  pgmname=VST2000 stepname=CLEANUP
ENDJOB
```

IF/THEN

VSE モードでの Sun MBM ジョブの実行においては、IF/THEN マクロは、条件およびその条件が真の場合に実行するアクションを定義します。

- 条件が真の場合、THEN 文で指定したアクションが実行されます。
- 条件が真でない場合、THEN 文で指定したアクションはバイパスされ、次の文で実行が再開されます。

形式

IF *condition* [*logical-operator condition*] [verbose]
THEN *action*

説明

condition [*logical-operator condition*]

logical-operator は AND または OR です。

条件は次のいずれかです。

MAXRC: 前に実行された全ステップのうちの最大リターンコード

LASTRC: 前のステップの最大リターンコード

pname: 記号パラメータの名前

比較演算子は次のいずれかです。

EQ: 等しい

NE: 等しくない

GT: より大きい

LT: より小さい

GE: 以上

LE: 以下

integer: 0 〜 4095 の 10 進整数

value: 別の記号パラメータの名前または 0 〜 50 文字の文字列。文字列は引用符で囲む必要がある

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

action

任意の有効な Sun MBM マクロを指定します。

注 – THEN 文は別の行に記述する必要があります。

例

```
...
EXECPGM pgmname=CIMS050 stepname=step0001
IF LASTRC NE 0
       THEN GOTO ERROR
...
```

```
LABEL ERROR
EXECPGM pgmname=CIMS999 stepname=step0005
ENDJOB
...
EXECPGM pgmname=CIMS050 stepname=step0001
IF MAXRC EQ 0
       THEN EXECPGM pgmname=CIMS000 stepname=normeoj
GOTO END_JOB
...
ENDJOB
...
IF PARMA EQ '\$PARMB' OR PARMA EQ 'ACCT'
       THEN EXECPGM pgmname=CIMS800 stepname=step0012
EXECPGM pgmname=CIMS850 stepname= step0013
...
ENDJOB
```

IF/THEN/ELSE/ENDIF

MVS JCL IF/THEN/ELSE/ENDIF マクロは、前のステップの状態に基づいてジョブステップの条件付き実行を可能にします。IF マクロ文の条件が真の場合、IF/THEN マクロ文と ELSE マクロ文の間の文が実行されます。IF 文の条件が偽の場合、ELSE マクロ文と ENDIF マクロ文の間の文が実行されます。

形式

```
IF [()] relational-expression[)] THEN
.
.
[ELSE]
.
.
ENDIF
```

注 – IF、THEN、および ENDIF は、すべて必須のキーワードです。

説明

relational-expression

relational-expression キーワード、比較演算子、論理演算子、および NOT 演算子を含めます。

キーワードは、次のとおりです。

RC: ステップのリターンコードをチェックする。RC は、ジョブで検出された最大のリターンコード値を参照する

stepname.RC: 指定されたステップのリターンコードをチェックする。stepname は、ジョブステップ名または現在のプロシージャーのステップのどちらかでなければならない

stepname.procstepname.RC: 指定されたプロシージャーステップ名のリターンコードをチェックする。stepname は、ジョブステップ名

ABEND: ジョブの前の不正終了状態をチェックする。ABEND 条件付きチェックおよび ONCONDCODE を使用することにより、不正終了発生後にステップを実行可能にできる

形式:

```
ABEND

ABEND=TRUE

stepname.ABEND
```

stepname.RUN: 前のステップが実行されたかをチェックする

stepname.procstepname.RUN: プロシージャーの特定ステップが実行されたかをチェックする

比較演算子:

EQ: 等しい

NE: 等しくない

GT: より大きい

LT: より小さい

NG: より大きくない

NL: より小さくない

GE: 以上

LE: 以下

論理演算子 AND または OR は、IF テストに対して複合条件を作ります。例を参照してください。

NOT 演算子はキーワード NOT であり、条件の反対をチェックします。次に例を示します。

```
IF step1.run THEN ...
```

とすると、step1 の実行が開始されている場合に真と評価されます。これに対し、

```
IF NOT step1.run THEN ...
```

とすると、step1 の実行が開始されなかった場合に真と評価されます。

例

STEP1 からのリターンコードが 8 より大きい場合に真となる基本的な IF 条件:

```
IF STEP1.RC GT 8 THEN
LABEL     name='ABRTSTP'
EXECPGM   pgmname='ABORT' stepname='ABRTSTP'
ELSE
LABEL     name='OKAY'
EXECPGM   pgname='PGM1' stepname='OKAY'
ENDIF
```

STEP1 からのリターンコードが 0 と等しく、かつ STEP1 が実行された場合に真となる複合 IF 条件:

```
IF STEP1.RC EQ 0 AND STEP1.RUN THEN
.
.
ELSE
.
.
ENDIF
```

複数の論理演算子がある複合条件。IF 条件は、左から右に向かって各組のオペランドをチェックします。最初に、*relational-expression* STEP1.RC EQ 0 AND STEP1.RUN の真偽をチェックします。次に、その結果と 2 番目の*relational-expression* STEP2.RC GT 4 の結果で論理和演算が行われ、IF 条件全体の最終評価となります。

```
IF STEP1.RC EQ 0 AND STEP1.RUN OR STEP2.RC GT 4 THEN
.
.
ELSE
.
.
ENDIF
```

ABEND *relational-expression* キーワードによる IF 条件:

```
IF ABEND THEN
.
.
ENDIF
```

LABEL

LABEL マクロは、ジョブまたはプロシージャーストリーム内で特定ポイントを定義します。これを GOTO マクロまたは subjob および unikixjob コマンドの -R および -S オプションとともに使用すると、後続の文をバイパスして指定されたラベルで実行を再開します。

形式

LABEL name=*label-name* [verbose]

説明

name=*label-name*

label-name は 1 ～ 8 文字の英数字です。*label-name* の最初の文字は英字である必要があります。必須の引数です。

label-name に記号パラメータを含めることはできません。LABEL マクロに指定できる *label-name* は 1 つだけです。

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

GOTO マクロで *label-name* を使用する場合、GOTO マクロは実行するジョブまたはプロシージャーの有効範囲内にあり、参照するラベルの前にある必要があります。

ジョブ内に GOTO *label-name* を指定する場合、そのジョブ内にターゲットラベルを定義する必要があります。

プロシージャー内に指定する場合、同じプロシージャー内にターゲットラベルを定義する必要があります。

例

GOTO マクロによる LABEL マクロの使用:

```
BEGINJOB
LABEL    name=STEP0001
ASSGNDD  ddname=INDET type=FS  \\\
         filename=/prod/inventory/detail
ASSGNDD  ddname=INSUM type=FS  \\\
         filename=/prod/inventory/summary
ASSGNDD  ddname=INVREPT type=SYSOUT
EXECPGM  pgmname=INV0001 stepname=STEP0001
IF LASTRC NE 0
      THEN GOTO ABENDJOB
EXECPGM  pgmname=INV0002
IF LASTRC EQ 0
      THEN GOTO GENRPT
LABEL ABENDJOB
EXECPGM  pgmname=AB9999
GOTO END_JOB
LABEL GENRPT
ASSGNDD  ddname=INSUM type=FS
ASSGNDD  ddname=INSUMREPT type=SYSOUT copies=2
EXECPGM  pgmname=RPT0001
ENDJOB
```

LASTRC

実行された最後のステップのリターンコードを取り出します。IF/THEN マクロとともに使用します。

形式

LASTRC

例

IF/THEN マクロによる使用:

```
BEGINJOB
EXECPGM pgmname=PAYROLL1 stepname=STEP001
IF LASTRC EQ 0
       THEN GOTO ENDJOB
EXECPGM pgmname=PAYERROR stepname=PAYABEND
ENDJOB
```

LIBDEF

実行可能プログラムまたはプロシージャーの検索パスを定義するFile_Map では、複数のファイルに対して同じ *datasetname* を指定できます。JOBCAT または STEPCAT DD 文によって指定された同じデータセットに対して複数の参照が存在することもありますが、それぞれの参照にはその File_Map エントリの *catalog-name* フィールドに別々のカタログ名があります。バッチシェルスクリプトの実行時に、Sun MBM は File_Map にアクセスして正確なパス名を取得します。

形式

LIBDEF [catalog=*catalog-name*] [concat=[Y|N]]
[datasetname=*datasetname*]
[lib='*dir*[:*dir*:...]']
[scope=JOB|STEP] type='*type*[,*type*...]' [verbose]

説明

catalog=

1 ～ 8 文字の英数字のカタログ名です。これは、ファイルおよびファイル属性を割り当てるときにシステムデフォルト (MASTERCAT) の前に検索するプライベートカタログの名前です。定義できるプライベートカタログは 1 つだけです。

concat=

現在の LIBDEF 文の lib オプションで定義されたディレクトリを既存のディレクトリと連結するかどうか指定します。concat=Y および type=PGM または PROC の場合、データセットに対して検索されるディレクトリが連結されます。引数は次のとおりです。

Y: lib 値を既存の検索パスに連結します。デフォルト

N: lib 値を既存の検索パスに連結しません。

注 – カタログ名の連結はサポートされていません。

---

datasetname=*datasetname*

メインフレーム上のデータセットのオリジナル名です。

lib=

プログラムおよびプロシージャーの場合、指定した検索パスに追加されるディレクトリまたはディレクトリのリストです。複数のディレクトリを定義するときは、コロンで区切ります。

プライベートカタログを定義する場合、*dir* はカタログ名を表します (複数のカタログ名はコロンで区切る)。プライベートカタログに存在可能なデータセットのエントリをサブシステムが組み込めるように、File_Map を変更する必要があります。File_Map については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

scope=

ライブラリ定義の有効範囲を定義します。

JOB: ジョブの終了まで、または後続の LIBDEF scope=JOB または LIBDROP マクロによって上書きされるまで、定義が有効。デフォルト

STEP: 現在のステップに対してだけ定義が有効

type=

1 つ以上の引数を選択します。複数の type を定義する場合、各パラメータをコンマで区切り、まとめてアポストロフィで囲む必要があります。

CAT: 指定したカタログ (1 ～ 8 文字の英数字) は、ファイルおよびファイル属性を割り当てるときにシステムデフォルト (MASTERCAT) の前に検索するプライベートカタログ。定義できるプライベートカタログは 1 つだけです。

PGM: 指定したディレクトリを実行可能プログラムに対するデフォルトの検索パスに追加する。デフォルトディレクトリは、サブシステム作成時に定義される

PROC: 指定したディレクトリをプロシージャー名に対するデフォルトの検索パスに追加する。デフォルトディレクトリは、サブシステム作成時に定義される

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

例

実行可能プログラムを検索する追加のディレクトリを定義します。

```
LIBDEF type=PGM lib='/user1/progs:/test/progs'
```

プログラムおよびプロシージャーを検索する追加のディレクトリを定義します。

```
LIBDEF type='PGM,PROC' lib='/user1/exec:/test/exec'
```

システムデフォルト MASTERCAT の前にデータセットを検索するプライベートカタログを定義します。

```
BEGINJOB
LIBDEF    scope=JOB type=CAT lib='TESTCAT'
...
LIBDEF    scope=STEP type=CAT lib='USER1CAT'
ASSGNDD   ddname=INFILE dataset=CUST.DATA
...
```

この例で、File_Map には次のものが含まれます。

```
CUST.DATA;MASTERCAT;FS;/prod/cust/data;;0;

CUST.DATA;TESTCAT;FS;/testcat/cust/data;;0;

CUST.DATA;USER1CAT;FS;/user1/cust/data;;0;
```

USER1CAT は検索するプライベートカタログとして現在のステップで定義された LIBDEF マクロによって定義されるので、割り当てられるファイルは /user1/cust/data です。scope=STEP で定義されたカタログは、現在のステップで scope=JOB の LIBDEF を上書きします。

LIBDROP

プロシージャーまたは実行可能プログラムへの検索パスをデフォルトディレクトリにリセットする LIBDEF scope=JOB マクロによって定義されたジョブレベルの検索パスを削除するために、このマクロを使用します。

注 – LIBDEF scope=*step* によって定義されたライブラリ検索パスは、ステップ終了時に自動的に削除されます。

形式

LIBDROP type='*type*[,*type*...]' [verbose]

説明

type=

1 つ以上の引数を選択します。複数の type を定義する場合、各パラメータをコンマで区切り、まとめてアポストロフィで囲む必要があります。

CAT: ファイルおよびその属性を割り当てるために ASSGNDD マクロを使用するときに、プライベートカタログへの参照を削除する。ASSGNDD マクロは、ファイルおよびその属性の割り当てのためにシステムマスターカタログ MASTERCAT だけを使用する

PGM: 実行可能プログラムへの検索パスをシステムデフォルトにリセットする。デフォルトの実行可能ディレクトリは、Sun MBM で定義される

PROC: プロシージャーへの検索パスをシステムデフォルトにリセットする。デフォルトのプロシージャーディレクトリは、Sun MBM で定義される

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

LIBLIST

プロシージャーまたは実行可能プログラムに対する現在の検索チェーンを表示する

形式

LIBLIST type='*type*[,*type*...]' [verbose]

説明

type=

1 つ以上の引数を選択します。複数の type= を定義する場合、各パラメータをコンマで区切り、まとめてアポストロフィで囲む必要があります。

CAT: データセットを割り当てるためのプライベートカタログの現在のパスを表示する

PGM: 実行可能プログラムに対する現在の検索パスを表示する

PROC: プロシージャーに対する現在の検索パスを表示する

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

LOGMSG

LOGMSG マクロは、Sun MBM ジョブの履歴ファイルに引用符付きテキストを書き込みます。

形式

LOGMSG '*comment*'

説明

comment

ジョブの履歴ファイルに書き込むテキストです。

注 – LOGMSG マクロで特殊文字を使用した場合の結果は予測できません。そのマクロ文を生成するとき、VSE JCL トランスレータはすべての特殊文字を下線に変換します。

DISPLAY および LOG については、『Sun Mainframe Batch Manager ソフトウェア移行ガイド』を参照してください。

MAXRC

直前までに実行された全てのジョブステップのうち、最大のリターンコードを返します。VSE JCL IF/THEN マクロとともに使用します。

形式

MAXRC

例

ステップ AR0003 を実行する前に、前のすべてのステップがリターンコード 0 で実行されたことを確認します。少なくとも 1 つのステップのリターンコートが 0 でなかった場合、AR0003 はバイパスされてステップ AR0004 が実行されます。

```
BEGINPROC  procname=ARMTDSUM
EXECPGM    pgmname=AR0001 stepname=AR0001
EXECPGM    pgmname=AR0002 stepname=AR0002
IF MAXRC EQ 0
      THEN EXECPGM pgmname=ARSUMM stepname=AR0003
GOTO EXITPROC
EXECPGM    pgmname=ARERROR stepname=AR0004
LABEL EXITPROC
ENDPROC
```

ONCONDCODE

後続ステップのいずれかで発行される条件コードに基づいて実行するアクションを定義します。デフォルトの条件コードはゼロです。ステップが異常終了すると、条件コードはゼロ以外の値にリセットされます。条件コードテストが真の場合、指定したアクションが実行されます。

形式

```
ONCONDCODE  comparator integer [operator comparator integer]
{BYPASS|CONTINUE|GOTO {label-name|stepname|END_JOB}}
[scope='JOB'|'STEP'] [verbose]
[poverride='y' [stepname='stepname']]
```

説明

comparator

大文字または小文字の比較タイプです。このエントリは定位置パラメータであり、必須です。

EQ: 等しい

NE: 等しくない

GT: より大きい

LT: より小さい

GE: 以上

LE: 以下

integer

ステップ条件コードを比較する 10 進整数。ジョブ開始時に、Sun MBM がデフォルト条件コードをゼロに設定します。ステップが異常終了すると、条件コードはゼロ以外の値に設定されます。このエントリは定位置パラメータであり、必須です。

operator

2 つの条件のテストを指定する論理演算子。このエントリは定位置パラメータであり、複数の *comparator*/*integer* を指定する場合だけ必須です。

AND: 両方の条件が真である場合に、定義済みアクションを実行する

OR: どちらか 1 つでも条件が真である場合に、定義済みアクションを実行する

BYPASS (MVS JCL)

現在のステップをバイパスします。

CONTINUE

現在のステップを実行します。

GOTO

次のうちいずれかを指定します。

label-name:

実行を再開するジョブまたはプロシージャーのポイントを定義します。これは LABEL マクロで指定した名前です。*label-name* は ONCONDCODE 文の後ろに指定する必要があります。

label-name は、同じジョブ制御レベルで定義する必要があります。ジョブにコーディングする場合、同じジョブ内に定義します。プロシージャーにコーディングする場合、同じプロシージャー内に定義します。

stepname:

実行を再開するジョブまたはプロシージャーのポイントを定義します。これは EXECPGM または EXECPROC マクロの *stepname* パラメータで指定した名前です。*stepname* は ONCONDCODE 文の後ろに指定する必要があります。(MVS JCL)

stepname は、同じジョブ制御レベルで定義する必要があります。ジョブにコーディングする場合、同じジョブ内に定義します。プロシージャーにコーディングする場合、同じプロシージャー内に定義します。

END_JOB:

比較が真の場合、ジョブを終了します。ENDJOB マクロで実行を再開します。

scope= (MVS JCL)

条件コード指示の有効範囲を次のように定義します。

JOB: テストおよびアクションをすべての後続ステップに適用する。デフォルト

STEP: テストおよびアクションを現在のステップにだけ適用する

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

poverride='y' (MVS JCL)

これはプロシージャーの上書きを示します。このオプションの後ろに stepname= オプションを設定しない場合、上書き条件はすべてのプロシージャーに対して有効となります。

stepname='*stepname*' (MVS JCL)

この条件が、次に呼び出されるプロシージャーの特定のステップの条件コードを上書きすることを示します。このオプションの前に、poverride='y' オプションを設定する必要があります。

VSE および MVS JCL の両方において、各ジョブの開始時に、Sun MBM は次のデフォルトの ONCONDCODE 指示を設定します。

```
ONCONDCODE NE 0 GOTO END_JOB
```

ONCONDCODE マクロは、グローバルな条件を設定します。この条件が設定されると、呼び出されたプロシージャーでローカルの指示が指定されない限り、呼び出されたすべてのプロシージャーで有効です。

呼び出されたプロシージャーで定義されているローカルの ONCONDCODE は、そのプロシージャーおよびそれが実行するプロシージャー内に限り、有効な ONCONDCODE 指示を上書きします。上位レベルのプロシージャーまたはジョブは、下位レベルプロシージャーの ONCONDCODE 指示を継承しません。

注 – ONCONDCODE 条件は、ONRETCODE 条件の前にテストされます。ONCONDCODE 条件が真の場合、指定したアクションが実行されます。

例

GOTO オプションによる ONCONDCODE 条件:

```
BEGINJOB
ASSGNDD ...
...
EXECPGM   pgmname=PROG001 stepname=STEP0001
ONCONDCODE GT 0 GOTO END_JOB
EXECPGM   pgmname=PROG002 stepname=STEP0002
ONCONDCODE EQ 255 GOTO ERRHANDLE
EXECPGM   pgmname=PROG003 stepname=STEP0003
GOTO END_JOB
LABEL ERRHANDLE
EXECPGM   pgmname=ERRHANDLE stepname=ERRHANDLE
ENDJOB
```

JOB 条件コードを定義する ONCONDCODE:

```
BEGINJOB
ONCONDCODE NE 0 BYPASS scope=JOB
...
```

次にプロシージャーの上書きの例を示します。

```
ONCONDCODE NE 0 CONTINUE scope='STEP' poverride='y' stepname='STEP1'
EXECPROC procname='PNAME' stepname='JSTP'
```

ONRETCODE

後続ステップによって発行されるリターンコードに基づいて実行するアクションを定義する

形式

ONRETCODE [MAXRC | LASTRC | *stepname* | *stepname.procstepname*]
comparator integer [*operator comparator integer*]
{BYPASS|CONTINUE|GOTO {*label-name*|END_JOB}}
[scope={'JOB'|'STEP'}] [verbose]
[poverride='y' [stepname='*stepname*']]

説明

MAXRC (MVS JCL)

前に実行された全ステップのうちの最大リターンコード

LASTRC (VSE JCL)

現在実行中のステップの直前に実行されたステップからのリターンコード

stepname (MVS JCL)

テストされるリターンコードを発行した前のジョブステップを指定します。指定したステップがプロシージャーにある場合、そのステップは同じプロシージャーにある必要があります。それ以外の場合、指定したステップはジョブステップであったことになり、プロシージャーにはありません。その指定したステップは、プログラム (EXECPGM マクロ) を実行したのであり、プロシージャー (EXECPROC マクロ) を呼び出したのではありません。

stepname.procstepname (MVS JCL)

前のジョブステップによって呼び出されたプロシージャーのステップを指定します。*stepname* は呼び出し元のジョブステップの名前を指定し、*procstepname* はテストされるリターンコードを発行したプロシージャーステップを指定します。*procstepname* によって指定されるステップは、EXECPGM マクロを含む必要があり、プロシージャーを呼び出しません。

注 – *stepname* または *stepname.procstepname* オプションを省略すると、コードは前の全ステップのリターンコードと比較されます。前のステップのいずれかで発行されたリターンコードによってテスト条件が真になると、定義済みアクションが実行されます。

comparator

大文字または小文字の比較タイプです。このエントリは定位置パラメータであり、必須です。

EQ: 等しい

NE: 等しくない

GT: より大きい

LT: より小さい

GE: 以上

LE: 以上

integer

リターンコードと比較する 0 ～ 4095 の 10 進整数であり、必須エントリです。

operator

2 つの条件のテストを指定する論理演算子。このエントリは定位置パラメータであり、複数の *comparator integer* を指定する場合だけ必須です。

AND: 両方の条件が真である場合に、定義済みアクションを実行する

OR: どちらか 1 つでも条件が真である場合に、定義済みアクションを実行する

BYPASS (MVS JCL)

条件が真である場合、現在のステップをバイパスします。

CONTINUE

比較が真の場合、ジョブの次の文で実行を再開します。

GOTO

次のうちいずれかを指定します。

label-name:

実行を再開するジョブまたはプロシージャーのポイントを定義します。これは LABEL マクロで指定した名前です。

label-name は、同じジョブ制御レベルで定義する必要があります。ジョブにコーディングする場合、*label-name* は同じジョブ内に定義します。プロシージャーにコーディングする場合、*label-name* は同じプロシージャー内に定義します。

END_JOB:

比較が真の場合、ジョブは不正終了し、Sun MBM がジョブ強制的な中止のメッセージを発行します。

scope= (MVS JCL)

リターンコード指示の有効範囲を次のように定義します。

JOB: テストおよびアクションをすべての後続ステップに適用する。デフォルト

STEP: テストおよびアクションを現在のステップにだけ適用する

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

poverride='y' (MVS JCL)

これはプロシージャーの上書きを示します。このオプションの後ろに stepname= オプションを指定しない場合、上書き条件はすべてのプロシージャーに対して有効となります。

stepname='*stepname*' (MVS JCL)

この条件が、次に呼び出されるプロシージャーの特定のステップの条件コードを上書きすることを示します。このオプションの前に、poverride='y' オプションを指定する必要があります。

VSE バッチジョブの場合、Sun MBM はジョブの開始時に次のデフォルト指示を動的に実行します。

```
ONRETCODE GE 16 GOTO END_JOB
```

デフォルト指示は、明示的な ONRETCODE 文により上書きできます。

scope=JOB パラメータがある ONRETCODE マクロは、グローバルな条件を設定します。この条件が設定されると、呼び出されたプロシージャーで上書きされない限り、呼び出されたすべてのプロシージャーで有効です。プロシージャーで上書きされた場合、そのプロシージャーおよびそれが実行するプロシージャーのみで有効です。上位レベルのプロシージャーまたはジョブは、下位レベルプロシージャーの ONRETCODE 指示を継承しません。

注 – ONCONDCODE 条件は、ONRETCODE 条件の前にテストされます。ONCONDCODE 条件が真の場合、指定したアクションが実行されます。

例

ONRETCODE が、ステップのリターンコードに基づいてジョブ実行フローを制御します。

```
BEGINJOB
ONRETCODE GE 4 OR LE 8 GOTO PRNTERR
EXECPGM   pgmname=PROG001 stepname=STEP0001
ONRETCODE GT 8 GOTO END_JOB
EXECPGM   pgmname=PROG002 stepname=STEP0002
GOTO END_JOB
LABEL PRNTERR
EXECPGM   pgmname=PRNTERR stepname=PRNTERR
GOTO END_JOB
LABEL ERRHANDLE
EXECPGM   pgm=ERRHANDLE stepname=ERRHANDLE
ENDJOB
```

ONRETCODE が特定のステップをバイパスします。

```
BEGINJOB
LABEL NAME=STEP0001
ASSGNDD...
ONRETCODE GE 8 BYPASS
EXECPGM   pgmname=PGM1 stepname=STEP0001
```

次に、プロシージャーのステップの上書きの例を示します。

```
ONRETCODE MAXRC NE 0 BYPASS scope='STEP' poverride='y' stepname='STEP1'
EXECPROC procname='PNAME' stepname='JSTP'
```

```
ONCONDCODE NE 0 CONTINUE scope='STEP' poverride='y' stepname='STEP1'
ONRETCODE MAXRC LT 0 BYPASS scope='STEP' poverride='y' stepname='STEP1'
EXECPROC procname='PNAME' stepname='JSTP'
```

PAUSE

PAUSE マクロは、ジョブの実行を中断します。ジョブが再開されるまで、ジョブの中断は継続します。ジョブの再開の詳細については、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

形式

PAUSE [msg='*comment*']

説明

msg='*comment*'

ジョブの履歴ファイルに書き込むテキストです。

例

```
BEGINJOB
...
EXECPGM   pgmname=PROG001 stepname=step0001
PAUSE msg='JOB SUSPENDED, MOUNT A/R MTD TAPE'
EXECPGM   pgmname=PROG002 stepname=step0002
...
```

SETDATE

SETDATE マクロは、ジョブ実行中の Sun MBM システム日付を上書きします。BEGINJOB マクロの後ろにコーディングします。後続の EXECPGM マクロによって実行されるすべてのアプリケーションプログラムに対して有効です。

形式

```
SETDATE value=MM/DD/YYYY [verbose]
```

説明

value=

現在の Sun MBM システム日付を上書きする新しい日付を指定します。

MM	月	01 ～ 12
DD	日	01 ～ 31
YYYY	年	1970 ～ 2030

verbose

マクロを冗長モードで実行します。環境変数の値およびその他の情報メッセージが、ジョブの履歴ファイルに書き込まれます。

注 – アプリケーションプログラムは、そのシステム日付の代わりに Sun MBM システム日付を取り出すことができます。詳細については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

例

ジョブ実行中の日付を上書きします。

```
BEGINJOB
SETDATE value=02/25/2004
...
EXECPGM   pgmname=PROG001 stepname=STEP0001
...
EXECPGM   pgmname=PROG002 stepname=STEP0002
ENDJOB
```

SETPARM

SETPARM マクロは、記号パラメータに値を割り当てます。次の構文を使用して、そのパラメータ値を取り出して後続のマクロ文に引数として渡せます。

'\$*parameter-name*'

次に例を示します。

```
SETPARM parms='PARM1=SYS0001'
EXECPGM pgmname=CB001,parm='\$PARM1'
```

形式

SETPARM parms='*parmname*=[*value*|LASTRC|MAXRC]
[,*parmname*=...][,*parmname*=...]' [verbose]

説明

parmname=

1 ～ 7 文字の英数字の記号パラメータの名前です。

value: 記号パラメータに値を割り当てる文字列。値を無効にするには、PARMA= のように値のない parm 名を入力する

LASTRC: 記号パラメータを前のステップのリターンコード値に設定する

MAXRC: 記号パラメータをジョブ内の前のステップによって発行されたジョブ条件コード値に設定する

verbose

マクロを冗長モードで実行します。環境変数の値およびその他のデバッグメッセージが、ジョブの履歴ファイルに書き込まれます。

例

parms= オプション:

```
SETPARM parms='GEN1=1,GEN2=,PARM1=MAXRC'
```

parms= オプションの連結:

```
SETPARM parms='SYS=prod,RUN=MTD' \\\
       parms='ADF=YES,MO=MAY'
```

SETPGMSW

SETPGMSW は、Micro Focus COBOL のプログラム可能なスイッチ COBSW に値を割り当てます。この値は、アプリケーションプログラムの Special-Names パラグラフで指定されるスイッチに対応します。スイッチパラメータについては、Micro Focus のマニュアルを参照してください。割り当てられた値は、後続のすべてのステップに対して有効です。

形式

SETPGMSW value=*string* [verbose]

説明

string

プログラム可能スイッチ COBSW のビット位置に対応する 9 文字の文字列。9 文字より少ない文字列を指定すると、その位置にデフォルトの X が Sun MBM によって設定され、それに対応するスイッチは変更されません。有効な文字は、次のとおりです。

0: COBSW の対応する位置をオフ (-) に設定

1: 対応する位置をオン (+) に設定

X: 対応する位置を変更しない

verbose

マクロを冗長モードで実行します。環境変数の値およびその他のデバッグメッセージが、ジョブの履歴ファイルに書き込まれます。

例

オフセット 0 のスイッチをオンに設定します。COBOL は 0 から始まるスイッチを参照します。0 から 8 までの 9 スイッチがあります。

```
SETPGMSW 1XXXXXXXX
```

実行時に、次の環境変数設定が生成されます。

```
COBSW=+0
```

オフセット 1 のスイッチを 0 に、オフセット 2 のスイッチを 1 に設定し、その他のすべてのビットを変更しません。

```
SETPGMSW X01XXXXXX
```

実行時に、次の環境変数設定が生成されます。

```
COBSW=X-1+2
```

この X は、スイッチの変更されない値を表します。

SETPRINT

SETPRINT マクロは、システム出力ファイルの生成時に参照される一連の印刷オプションを定義します。システム出力ファイル (SYSOUT ファイル) は、ASSGNDD マクロの type=SYSOUT オプションを指定することによって生成されます。印刷オプションを有効にするには、SETPRINT マクロを参照する ASSGNDD マクロの前にそのマクロを置く必要があります。

プロシージャーに定義する SETPRINT マクロは、BEGINPROC マクロの後ろに置く必要があります。ASSGNDD マクロおよび SETPRINT マクロによる印刷制御の方法については、73 ページの「印刷出力の管理」を参照してください。

SETPRINT マクロは、 JES2 /*OUTPUT 文のキーワードパラメータとして使用される IBM 1 文字エイリアスもサポートします。たとえば、DEST= の代わりに D= を使用できます。

形式

```
SETPRINT [ddname='external-file-name'] printid='label'
[scope='JOB'|'STEP'|'JOB_DEFAULT'|'STEP_DEFAULT'|'DD']
[overridedd='Y'|'N']
[sysout-attribute='value' [sysout-attribute='value'...]][verbose]
```

説明

ddname=

各属性が適用される ASSGNDD マクロの ddname オプションに割り当てられる値。このオプションによって、後続 ASSGNDD マクロは SETPRINT マクロで定義された印刷オプションを継承できます。scope='DD' を使用する場合だけ、必須で有効なエントリです。

printid=

このパラメータは必須値は 1 ～ 8 文字の英数字の文字列です。指定した一連の印刷オプションに関連付けられているラベルを定義します。

scope=

属性を継承できる SYSOUT ファイルを定義します。SYSOUT ファイルは、ASSGNDD マクロの type=SYSOUT オプションを指定することによって生成されます。

SETPRINT マクロの一連の属性への明示的参照は、SYSOUT ASSGNDD マクロの printid=*label* オプションによって行われます。有効範囲を JOB_DEFAULT または STEP_DEFAULT に定義する場合、明示的参照は必要ありません。

JOB: ジョブのすべての SYSOUT ファイルに対する、SETPRINT マクロで定義される印刷属性。その SETPRINT マクロを printid によって明示的に参照する SYSOUT ASSGNDD マクロだけが、オプションを継承する

STEP: SETPRINT マクロで定義される印刷属性が、現在のステップのすべての SYSOUT ASSGNDD ファイルで使用可能。その SETPRINT マクロを printid によって明示的に参照する SYSOUT ASSGNDD マクロだけが、オプションを継承する

JOB_DEFAULT: SETPRINT マクロで定義される印刷オプションが、ジョブによって生成されるすべての印刷出力ファイルに適用される。すべての SYSOUT ASSGNDD ファイルが、別の SETPRINT マクロを明示的に参照しない限り、属性を継承する

STEP_DEFAULT: SETPRINT マクロで定義される印刷オプションが、ステップによって生成されるすべての SYSOUT ASSGNDD ファイルに適用される。ステップのすべての SYSOUT ASSGNDD ファイルが、別の SETPRINT マクロを明示的に参照しない限り、属性を継承する

DD: デフォルトSETPRINT マクロで定義される印刷オプションが、ddname=*external-file-name*によって指定される SYSOUT ASSGNDD ファイルだけに適用される。このパラメータには ddname= オプションが必須

overridedd=

N: デフォルト SETPRINT およびそのオプションを継承する ASSGNDD マクロで同一の印刷オプションを定義すると、ASSGNDD オプションの値が SETPRINT の値を上書きする

注 – ASSGNDD マクロで使用可能な印刷オプションは、dest= および copies= だけです。

Y: SETPRINT およびそのオプションを継承する ASSGNDD で同一の印刷オプションを定義すると、SETPRINT オプションの値が ASSGNDD の値を上書きする

sysout-attribute=

システムの OUTPUT ファイルに対して定義する印刷オプション。ASSGNDD マクロの type=SYSOUT オプションによって作成されるファイルに適用されます。環境変数が実行時に設定され、選択したスプール処理方法によって使用されて印刷を制御します。構文および上書き動作については、73 ページの「印刷出力の管理」を参照してください。

verbose

マクロを冗長モードで実行します。環境変数の値およびその他のデバッグメッセージが、ジョブの履歴ファイルに書き込まれます。

SETRETCODE

現在のステップのリターンコードをユーザー作成ユーティリティーから指定される数値に設定します。

形式

SETRETCODE *nnnn*

説明

nnnn

数値リターンコード。

例

ユーザーユーティリティーを実行し、エラーの発生を検出するジョブステップ。

ユーザーユーティリティー USRUTIL1:

```
#!/bin/ksh
echo "start USRUTIL1"
if[-f${SEQFILES}/custupdt/tran5]
then
...
...
else
 echo "USRUTIL1 (E) files does not exist: ${SEQFILES}/custupdt/tran5"
 SETRETCODE 12
```

ファイル tran5 が見つからない場合、ユーティリティー USRUTIL1 は SETRETCODE 12 を実行し、リターンコードを 12 に設定します。これにより STEP0002 はバイパスされ、STEPERROR が実行されます。そのあと、ジョブは終了します。

```
BEGINJOB
ONRETCODE GE 12 GOTO STEPERROR scope=JOB
#-----------------------------------------------------------------
LABEL    stepname=STEP0001
EXECPGM  pgmname=USRUTIL1
#-----------------------------------------------------------------
LABEL    stepname=STEP0002
ONRETCODE GT 8 GOTO END_JOB scope=STEP
EXECPGM  pgmname=COBOL1
#-----------------------------------------------------------------
LABEL    stepname=STEPERROR
EXECPGM  pgmname=ERROR
ENDJOB
```

印刷出力の管理

この節では、Sun MBM マクロ環境での印刷出力の管理方法について説明します。出力ファイルは、ASSGNDD マクロの type=SYSOUT オプションを指定することによって作成されます。作成されたファイルは、SYSOUTDIR 環境変数によって定義されるシステム出力ディレクトリに入れられます。印刷を制御する属性は、BEGINJOB、ASSGNDD、および SETPRINT マクロの SYSOUT 属性オプションによって設定し、選択したスプール処理機能によって使用されて印刷を管理します。

次の小節では、サポートされるオプションおよび構文について、さらに ASSGNDD および SETPRINT マクロで、または BEGINJOB マクロでオプションを指定する場合の上書きルールについて説明します。

SYSOUT 属性

システム出力ファイルの印刷および処理を制御する SYSOUT 属性は、SETPRINT マクロで指定します。これらの属性は、各ステップの終了時に印刷スプール処理インタフェースによって使用される環境変数を設定します。

環境変数割り当ての書式は、表 3-2 に示します。*ddname* が参照される場合、ASSGNDD マクロの ddname=*external-file-name* オプションによって指定された *external-file-name* が参照されます。

$PUBLIC/bin/post_exec_pgm または post_exec_pgm.vse ファイルは、各ステップの終了後に実行されます。Sun MBM 管理者は、このファイルを設定してスプール処理パッケージとのインタフェースをとることができます。SYSOUT 関連属性に対して設定される環境変数が、そのスクリプトで使用可能です。post_exec_pgm 文の使用方法については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』 を参照してください。メインフレームのオペレーティングシステム環境から移行する場合、その環境の情報を参照する必要があります。

形式

[*sysout-attribute*='*value*'...]

印刷関連の指示を定義します。

説明

value は指示に関して選択するオプションを指定します。次の表に示すオプションを 1 つ以上指定します。

表 3-2 SYSOUT 属性パラメータのオプション (1 / 5)

オプション	環境変数の書式
burst={Y|N} 出力を別々の用紙にバーストするか、連続折りたたみ用紙に印刷する。 Y: 出力をバーストする N: 出力をバーストしない	*ddname*_BURST=Y *ddname*_BURST=N
chars=DUMP chars=*table-name* chars='*table-name*,*table-name*,...' SYSOUT ファイルの印刷に使用したり、文字のダンプを生成したりするための文字配列テーブルの名前	*ddname*_CHARS DUMP *ddname*_CHARS table-name *ddname*_CHARS '*table-name*,*table-name*'
ckptline=*number-lines* SYSOUT ファイル印刷時に論理ページに含める最大行数	*ddname*_CKPTLINE *number-line*
ckptpage=*number-pages* SYSOUT ファイル印刷時にチェックポイントをとるまで印刷する最大ページ数	*ddname*_CKPTPAGE *number-pages*
ckptsec=*number* SYSOUT ファイル印刷時にチェックポイントをとる間隔の秒数	*ddname*_CKPSEC *number*
class=*class-id* sysout データセットを指定した出力クラスに割り当てる。 ASSGNDD マクロの値が、SETPRINT マクロのクラスを上書きする。この動作は、SETPRINT マクロに overridedd=Y オプションをコーディングすると逆になる。 SETPRINT と ASSGNDD の両方が、BEGINJOB マクロに指定された jobclass を上書きする	*ddname*_CLASS *class-id*
compact=*table-name* SYSOUT ファイルの処理時に使用する圧縮テーブル	*ddname*_COMPACT *table-name*

表 3-2　SYSOUT 属性パラメータのオプション (2 / 5)

オプション	環境変数の書式
control=*spacing* SYSOUT ファイルの行間 *spacing* は次のいずれかを指定する。 PROGRAM | SINGLE | DOUBLE | TRIPLE	*ddname*_CONTROL *spacing*
copies=*number* copies='*group-val*,*group-val*,...' SYSOUT ファイルの印刷部数。3800 プリンタでファイルを印刷する場合、1 つ以上のグループ値の入力が可能。ASSGNDD マクロを使用してこのオプションを設定することもできる。 SETPRINT マクロの値が、ASSGNDD マクロの値を上書きする。デフォルト この動作は、SETPRINT マクロに overridedd=Y オプションを使用することにより変更できる	*ddname*_COPIES *number* *ddname*_COPIES *grp-val* *ddname*_COPIES *'group-val,group-val'*
datack=*directive* レポートする印刷エラーのタイプ *directive* は次のいずれかを指定する。 BLOCK | UNBLOCK | BLKCHAR | BLKPOS	*ddname*_DATACK *spacing*
dest=*dest-name* dest='*node*,*userid*' SYSOUT ファイルの出力先。ASSGNDD マクロを使用してこのオプションを設定することもできる。 ASSGNDD マクロの値が、SETPRINT マクロの値を上書きする。デフォルトの動作。この動作は、SETPRINT マクロに overridedd=Y オプションを使用することにより変更できる	*ddname*_DEST *dest-name* *ddname*_DEST *'node,userid'*
fcb=*fcb-image-id* fcb='*fcb-image-id*,VERIFY' fcb='*fcb-image-id*,ALIGN' FCB 機能があるプリンタに対して読み込む用紙制御バッファー。FCB イメージは、特定プリンタでの用紙の動きを制御する	*ddname*_FCB *number* *ddname*_FCB_STATUS VERIFY *ddname*_FCB_STATUS ALIGN
flash=*overlay-id* flash='*overlay-id*,*count*' flash=NONE *overlay-id*: SYSOUT ファイル印刷時に使用する書式オーバーレイ *count*: オーバーレイを印刷する部数	*ddname*_FLASH *overlay-id* *ddname*_FLASH *'overlay-id,count'* *ddname*_FLASH NONE
formdef=*member-name* オーバーレイ書式の使用を制御する、ライブラリに定義されるメンバー名	*ddname*_FORMDEF *form-identifier*

表 3-2　SYSOUT 属性パラメータのオプション (3 / 5)

オプション	環境変数の書式
forms=*form-identifier* forms=STD SYSOUT ファイルに印刷する用紙の識別子	*ddname*_FORMS *form-identifier* *ddname*_FORM *form-identifier* *ddname*_FORMS STD *ddname*_FORM STD 書式 *ddname*_FORM は、後方互換性のために設定される
free={END|CLOSE} SYSOUT ファイルに関連付けられたリソースを割り当て解除するときを指定する。 END: ジョブステップ終了時に割り当て解除する。デフォルト CLOSE: プログラムが SYSOUT ファイルを閉じるときに割り当て解除する	*ddname*_FREE END *ddname*_FREE CLOSE
groupid=*group-identifier* SYSOUT ファイルが、特定の出力グループに関連付けられる。出力グループは、クラス、出力先などの同様の特性を持つ	*ddname*_GROUP *group-identifier*
hold=*directive* 後続コマンドによって解放されるまで、印刷対象の SYSOUT ファイルをシステムが保持する。 Y|YES: 解放されるまでファイルを保持する N|NO: 印刷のためにファイルを解放する	*ddname*_HOLD YES *ddname*_HOLD NO
index=*number* SYSOUT ファイルの左マージンをインデントする位置	*ddname*_INDEX *number*
jesds=*message-type* SETPRINT マクロによって定義されたオプションで処理するジョブの一覧およびログメッセージのタイプ。 *message-type* は次のいずれかを指定する。 ALL | JCL | LOG | MSG	*ddname*_JESDS *message-type*
lindex=*number* SYSOUT ファイルの右マージン	*ddname*_LINDEX *number*
linect=*number* 1 ページの最大行数	*ddname*_LINECT *number*

表 3-2　SYSOUT 属性パラメータのオプション (4 / 5)

オプション	環境変数の書式
modify=*module-name* modify='*module-name*,*number*' *module-name*: SYSOUT ファイルの印刷方法を制御するために使用するコピー変更モジュール *number*: 文字セット名のリストを SETPRINT マクロの chars= オプションによって定義する場合、SYSOUT ファイルの説明文および列見出しを定義するために使用する文字セットのリストのオフセット	*ddname*_MODIFY *module-name* *ddname*_MODIFY '*module-name,number*'
pagedef=*module-name* ページモードプリンタで SYSOUT ファイルを印刷するときに使用するモジュール	*ddname*_PAGEDEF *module-name*
pimsg=*directive* pimsg='*directive*,*number*' 出力データセットの終了時に印刷エラーメッセージをシステムが出力するか、抑止するか *directive* は次のいずれかを指定する。 YES: エラーメッセージを出力する NO: エラーメッセージを抑止する *number*: SYSOUT ファイルの印刷を停止するまでに出力するエラーの最大数	*ddname*_PIMSG *directive* *ddname*_PIMSG '*directive,number*'
prmode=*directive* SYSOUT ファイルの印刷のモード *directive* は次のいずれかを指定する。 LINE: ラインモードプリンタへのスケジュール PAGE: ページモードプリンタへのスケジュール *mode*: 選択したモードの識別子 (1 ～ 8 文字の英数字)	*ddname*_PRMODE *directive*
prty=*number* 出力キューでの SYSOUT 優先順位	*ddname*_PRTY *number*
threshld=*number* SYSOUT ファイルの最大サイズ	*ddname*_THRESHLD *number*

表 3-2　SYSOUT 属性パラメータのオプション (5 / 5)

オプション	環境変数の書式
trc=*record-status* SYSOUT ファイルの各論理レコードにテーブル参照文字 (TRC) コードを含めるかを指定する *record-status* は次のいずれかを指定する。 Y: TRC コードを含める N: TRC コードを含めない	*ddname*_TRC Y *ddname*_TRC N
ucs=*charset-identifier* *charset-identifier* SYSOUT ファイルの印刷に使用する汎用文字セットの名前	*ddname*_UCS *charset-identifier*
writer=*writer-name* SYSOUT ファイルを処理するために読み込む外部ライターモジュール	*ddname*_WRITER *module-name* *ddname*_REPORT *module-name* 書式 *ddname*_REPORT writer-**名前** は、後方互換性のために設定される

SYSOUT 属性の上書き

ASSGNDD マクロとそれが参照する SETPRINT マクロの両方で、dest および copies 属性を定義できます。SYSOUT 属性を処理するときの上位から下位へのデフォルトの優先順位は、次のとおりです。

- ASSGNDD マクロでコーディングされた属性
- printid オプションを介して ASSGNDD マクロによって明示的に参照される SETPRINT マクロにコーディングされた属性
- ステップの ASSGNDD によって暗黙的に参照される SETPRINT scope='STEP_DEFAULT' マクロにコーディングされた属性
- ジョブの ASSGNDD によって暗黙的に参照される SETPRINT scope='JOB_DEFAULT' マクロにコーディングされた属性

SETPRINT マクロの overridedd='Y' オプションが指定されている場合、SETPRINT 属性が ASSGNDD マクロの対応する属性を上書きします。

例

ジョブによって作成されるすべての SYSOUT ファイルに適用される印刷オプションを定義します。printid オプションは必須ではありません。JOB_DEFAULT SETPRINT マクロによって定義される印刷オプションは、SYSOUT ファイルを定義するすべての ASSGNDD マクロに適用されます。SETPRINT と ASSGNDD の両方に同じ値がある場合、ASSGNDD に指定されている値が優先指定になります。

```
BEGINJOB
SETPRINT  scope='JOB_DEFAULT' dest='LOC5' copies='3' form='FRM1'

#STEP0001-----------------------------------------------------

ASSGNDD   ddname=REPORT1 type=SYSOUT dest=RMT1 copies=2  \\\
          disp=o normal=k abend=k
ASSGNDD   ddname=REPORT2 type=SYSOUT disp=o normal=k abend=k
# When the following program, PRNT001, is executed the
# print options in effect are:
#
# for ddname REPORT1: dest=RMT1 copies=2 form=FRM1
# for ddname REPORT2: dest=LOC5 copies=3 form=FRM1
EXECPGM   pgmname=PRNT001 stepname=STEP0001
ENDJOB
```

SYSOUT ファイルに対する ASSGNDD マクロがその ASSGNDD マクロの printid オプションによって SETPRINT を明示的に参照する場合、ジョブによって作成される SYSOUT ファイルに適用される印刷オプションを定義します。ASSGNDD マクロには、SETPRINT のオプションを継承するために printid オプションが必要です。

```
...
SETPRINT  printid='RMT0' scope='JOB' dest='LOC5' copies='3' form='FRM0'
#STEP0001-----------------------------------------------------------
# ddname REPORT1 explicitly references the SETPRINT macro
ASSGNDD   ddname=REPORT1 type=SYSOUT printid=PRNT1 dest=RMT1 \\\
          disp=o normal=k abend=k
# ddname REPORT2 does not refer to the SETPRINT macro
ASSGNDD   ddname=REPORT2 type=SYSOUT form=FRM2 copies=2       \\\
          disp=o normal=k abend=k
# When the following program,PRNT001,is executed the print
# options in effect are:
# for ddname REPORT1: dest=RMT1 copies=3 form=FRM0
# for ddname REPORT2: form=FRM2 copies=2
EXECPGM   pgmname=PRNT001 stepname=STEP0002
ENDJOB
```

ステップ内で作成されるすべての SYSOUT ファイルに適用される印刷オプションを定義します。ASSGNDD マクロで printid オプションが必要です。SETPRINT マクロ、scope='STEP' によって定義される印刷オプションが、現在のステップ内の SYSOUT ファイルを定義する ASSGNDD マクロにだけ適用されます。

```
BEGINJOB
...
SETPRINT  printid=PT01 scope=STEP dest=LOC5 copies=3 form=FRM1

ASSGNDD   ddname=PRNTFIL2 type=SYSOUT printid=PT01 dest=RMT1 \\\
          disp=o normal=k abend=k
#-------------------------------------------------------------------
# When the following program, PRNT002, is executed,
# the options in effect are:
# for ddname PRNTFIL2 dest=RMT1 copies=3 form=FRM1
#-------------------------------------------------------------------
EXECPGM   pgmname=PRNT002 stepname=STEP0002
...
ENDJOB
```

SETPRINT マクロの overridedd オプションを使用します。SETPRINT マクロで overridedd='Y' オプションが指定されている場合、SETPRINT で指定されている値が ASSGNDD で指定されている値を上書きします。

```
BEGINJOB
SETPRINT  scope='JOB_DEFAULT' dest='LOC5' copies='3' form='FRM1' overridedd='Y'
#STEP0001-----------------------------------------------------------
ASSGNDD   ddname=REPORT1 type=SYSOUT dest=RMT1 copies=2 disp=o \\\
          normal=k abend=k

ASSGNDD   ddname=REPORT2 type=SYSOUT disp=o normal=k abend=k
# When the following program, PRNT001, is executed the
# print options in effect are:
#
# for ddname REPORT1: dest=LOC5 copies=3 form=FRM1
# for ddname REPORT2: dest=LOC5 copies=3 form=FRM1
EXECPGM   pgmname=PRNT001 stepname=STEP0001
ENDJOB
```

付録 A

特殊文字の変換

この付録では、Sun MBM での特殊文字の処理方法について説明します。

特殊文字を使用してマクロをコーディングする場合は、次の書式に従う必要があります。

/x*nn*

この *nn* は、ASCII 文字の 16 進表記です。

JCL に含まれる特殊文字は、トランスレータによって ASCII 文字の 16 進表記に変換されます。たとえば、`$$` は `/x24/x24` に変換されます。

次の表は、特殊文字との対応を示します。

表 A-1　特殊文字

文字	マクロ文での使用
\	\\
!	\\!
#	\\#
“	/x22
$	/x24
:	/x3A
;	/x3B
<	/x3C
>	/x3E
?	/x3F
&&	/x26
‘’	/x27

表 A-1　特殊文字 (続き)

文字	マクロ文での使用
(	/x28
)	/x29
*	/x2A
[	/x5B
]	/x5D
^	/x5E
|	/x7C
~	/x7E

用語集

A

Animator (名詞) Micro Focus COBOL のソースレベルデバッガ。

B

Batch Administration Manager (BAM) (名詞) Sun MBM ノードやサブシステムを設定したり管理したりするために使用されるツール。

`batchenv` ファイル (名詞) ノードの実行方法を制御する環境変数を含む設定ファイル。各ノードは固有の `batchenv` ファイルを持ち、それらを実行しないとノードが開始されない。

`bqm` 「バッチキューマネージャー」を参照。

D

`dostrans` (名詞) VSE JCL トランスレータ。

E

ebmmd　「メッセージデーモン」を参照。

F

File_Map　(名詞) IBM データセット、ライブラリ、および世代別データグループ (GDG) と、対応する UNIX パス名とを関連付けるエントリを含む特別ファイル。このファイルは、Sun MBM JCL トランスレータおよびサブシステムによって、メインフレーム JCL ストリームの変換時やマクロジョブスクリプトの実行時に使用される。サブシステムはそれぞれ 1 つの File_Map に関連付けられている。

I

.install ファイル　(名詞) インストールされたノードおよびそれに関連付けられたサブシステムすべての情報を含んだグローバル Sun MBM 設定ファイル。.install ファイルは、ノードをインストールディレクトリにインストールするときに作成される。.install ファイルは、ノードを開始するたびに読み込まれる。

J

jon　(名詞) ジョブのオカレンス番号。ジョブがサブミットされると、Sun MBM は一意の 3 桁のオカレンス番号を割り当てる。*jon* は多くのコマンドに参照される。

K

KIXSYS　(名詞) システムテーブルが位置する Sun MTP 領域のディレクトリをポイントする環境変数。Sun MBM はこの値を使用して領域に接続する。

M

mvstrans (名詞) MVS JCL トランスレータ。

P

POWER ジョブのスケジューリング、入出力のスプール処理、パーティションの開始および終了を管理する VSE オペレーティング環境のコンポーネント。

psg_daemon 「プロセスグループデーモン」を参照。

S

Sun Mainframe Batch Manager ソフトウェア (Sun MBM) (名詞) 制御された環境でバッチジョブを実行するための機能を提供するバッチマネージャー製品。Sun MBM は、バッチ生産負荷を処理し、開始時刻やバッチプロセスの最大数、およびジョブの優先順位といった割り当てられたパラメータによってジョブをスケジューリングする。

Sun Mainframe Transaction Processing ソフトウェア (Sun MTP) (名詞) プロセス間通信サービス、ソケット、および COBOL、C、PL/I などの機能を使用してアプリケーションを実行するユーザーアプリケーション。クライアント以外の Sun MTP のすべてのコンポーネントは、メインサーバープロセスである `unikixmain` によって起動する。

Sun MTP 領域 (名詞) システム上の異なるアプリケーションを定義するプロセス変数、リソース変数、および環境変数のセット。

SYSIN ファイル (名詞) システム入力ファイル。このファイルには、サブミットされたジョブのうち実行待ちのものすべてが格納される。

V

VSAM 構成テーブル (VCT) (名詞) 基本の Sun MTP 構成パラメータを定義する制御テーブル。Sun MTP 領域を Sun MBM ノードに接続するとき、このテーブルはノードのインストールディレクトリを含んでいる必要がある。

あ

アクティビティー (名詞) ジョブを実行する共有メモリのセグメント。アクティビティーはジョブクラスに割り当てられる。1 つのクラスに 99 までのアクティビティーを割り当てることができる。

アテンションルーチン (AR) (名詞) 文の最初の 2 つの位置に `//` なしに `SYSLOG` (コンソール) から入力され、ジョブストリームにはない VSE JCL 文。

アニメートする (動詞) COBOL ソースレベルデバッガを呼び出すこと。

え

エラー表 (名詞) Solaris オペレーティングシステムに付属するインクルードファイルのこと。このファイルへのアクセス方法が分からない場合は、システム管理者にお問い合わせください。

か

仮想記憶アクセス方式 (VSAM) (名詞) さまざまなアクセス方式によってレコードにアクセスする方式。

ESDS (入力順データセット)。レコードは順次に記録され、アクセスされる。

RSDS (相対レコードデータセット)。レコードは、データセット内で占める位置番号によって検索される。

KSDS (キーシーケンスデータセット)。レコードは索引またはキーによって検索される。

仮想コンソール機能 (vcf) (名詞) バッチジョブ実行によるすべての出力メッセージを処理するデーモン (`vcf`)。

環境変数 (名詞) プログラムファイルおよびアプリケーションの位置を定義する変数。クライアントとサーバーは、どちらも環境変数を使用する。

き

許可ファイル (名詞) ユーザーの、ノードを開始、管理、停止する権限、サブシステムを管理する権限、およびクラスやアクティビティーを作成、変更、削除する権限を制御するファイル。

く

クラス (名詞) 1 つまたは複数のアクティビティーを含む概念エンティティー。ジョブはクラスにサブミットされ、アクティビティーが利用可能な場合、ジョブは実行される。1 つのノードで 26 クラスをサポートできる。

け

検査する (動詞) MVS または VSE トランスレータのどちらかを使用して JCL ジョブやプロシージャーを実行し、ファイルやプログラムがすべて存在するか検査すること。検査モードでジョブやプロシージャーを実行すると、`File_Map` が 1 つ作成される (既にファイルが存在している場合は、追加される)。

こ

コンソール端末 (名詞) ノードにオペレータコンソールとして定義される端末デバイス。さまざまなタイプのエラーメッセージを設定して、コンソール端末に表示できる。

コンソールファイル (名詞) Sun MBM コンソール端末に表示されるものと同じメッセージタイプを格納する連続したファイル。

さ

サブシステム (名詞) 特定のノードに従属する環境であり、ここで特定のタイプのジョブが実行される。たとえば、Sun MTP 領域の VSAM データセットにアクセスするジョブを実行するために使用するサブシステムを作成することができる。

サブシステム設定ファイル (名詞) サブシステムの作成時、BAM は 2 つの設定ファイルを作成する。1 つは読み取り専用ファイルで、サブシステム作成時に設定される環境変数を含んでいる。もう 1 つはユーザー編集可能ファイルで、他の環境変数を追加できる。これらのファイルはそれぞれ、`$SETUP` および `$USER_SETUP` として参照される。

し

シグナル表 (名詞) Solaris オペレーティングシステムに付属するインクルードファイルのこと。このファイルへのアクセス方法が分からない場合は、システム管理者にお問い合わせください。

ジョブエントリ制御言語 (JECL) (名詞) ジョブ入力、文の収集とスケジューリング、および出力スプール処理を管理するための POWER JCL ストリームで使用される VSE JCL 文。

ジョブクラス 「クラス」を参照。

ジョブ制御コマンド (JCC) (名詞) 文の最初の 2 つの位置に // なしにコーディングされ、ジョブストリームにありうる VSE JCL 文。

ジョブ制御文 (JCS) (名詞) 文の最初の 2 つの位置が // でコーディングされ、ジョブストリームにある VSE JCL 文。

ジョブの順序付け (名詞) ジョブを特定の順番で実行する機能。

ジョブの同期 (名詞) ジョブ実行の順番を制御する機能。ジョブの同期は、ジョブ実行の依存関係を設定できる点で、ジョブの順序付けとは区別される。たとえば、JOBC を実行する前に、JOBA および JOBB の両方を完了する必要がある。

ジョブリスト出力ファイル (名詞) ジョブの履歴情報を含むファイル。

す

スレッド 「アクティビティー」を参照。

せ

世代別データグループ (GDG) (名詞) 同じデータセット名を使用し、年代順に関連付けられたデータセットの集合。

つ

ツールキット (名詞) グラフィカルユーザーインタフェースを持つ Sun MBM アプリケーションで、これによりユーザーは File_Map を管理したり、JCL ジョブやプロシージャーを検査および変換したり、COBOL プログラムをコンパイルしたりできる。

て

デフォルトのサブシステム　(名詞) ジョブのサブミット時にサブシステムが指定されていない場合に、ジョブが実行されるサブシステム。

の

ノード　(名詞) Sun MBM ソフトウェアの一意のインストール。

は

バッチキューマネージャー　(名詞) Sun MBM サブシステムにサブミットされたジョブを管理するデーモン (`bqm`)。

バッチシェル　(名詞) C シェル環境のスーパーセットである Sun MBM バッチ実行環境。

ふ

ファイルシステム　(名詞) 物理ディスクドライブをパーティションと呼ぶ小単位の領域に分割する機能。パーティションには、ファイルシステム、スワップ空間、ブートセクタその他の情報を含めることができる。

ファイルのアクセス権 (またはモード)　(名詞) オペレーティングシステムの定義に従って、ファイルへのアクセスを制御する。

プロジェクト　(名詞) ジョブとプロシージャーのユーザー定義グループであるジョブエディタ構成概念。

プロセスグループデーモン　(名詞) `bqm` デーモンから要求を受け取り、実行されるバッチジョブについての情報を取得したり、バッチジョブをサブミットしたりするデーモン。

へ

変換する (動詞) MVS または VSE トランスレータのどちらかを使用して JCL ジョブやプロシージャーを実行し、ジョブやプロシージャーのマクロ文を作成すること。

ま

マクロ文 (名詞) MVS または VSE JCL ジョブおよびプロシージャーの変換によって生成される文。

マニュアルページ (名詞) `man` コマンドを使用して、コマンドの使用方法を表示できる。たとえば、`grep` コマンドについて表示するときは、プロンプトで `man grep` と入力する。

め

メッセージデーモン (名詞) Sun MBM プロセスのメッセージ交換サーバーとして機能するデーモン (`ebmmd`)。

ゆ

優先順位 (名詞) ジョブがサブミットされると、明示的または暗黙的に優先順位が割り当てられる。有効な値は `0` ～ `9` までで、`9` が最優先となる。

り

履歴ファイル (名詞) ジョブの開始およびジョブの終了メッセージを含む循環ファイル。

る

ルートファイルシステム　(名詞) オペレーティングシステムと関連のファイルが入っている。ルートファイルシステムは、完全なファイル名の最初の文字としてスラッシュ (/) をつけて、参照される

索引

き

く

け

こ

さ

し

す

せ