(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2010-73015
(P2010-73015A)
(43)公開日　平成22年4月2日(2010.4.2)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 9/54	(2006.01)	G06F	9/46　480B	

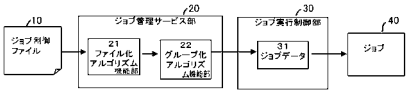
審査請求 未請求　請求項の数 10　ＯＬ　（全 18 頁）

(21)出願番号　特願2008-241022 (P2008-241022)
(22)出願日　平成20年9月19日 (2008.9.19)

(71)出願人　000152985
株式会社日立情報システムズ
東京都品川区大崎１－２－１
(74)代理人　100074550
弁理士　林　實
(74)代理人　110000073
特許業務法人プロテック
(72)発明者　原園　拓伸
東京都品川区大崎１－２－１　株式会社日立情報システムズ内
(72)発明者　上田　真
東京都品川区大崎１－２－１　株式会社日立情報システムズ内

(54)【発明の名称】ジョブ管理システム及びジョブ管理方法

(57)【要約】
【課題】処理属性を付与したジョブ制御ファイルの生成。
【解決手段】ジョブ管理サービス機能部２０のファイル化アルゴリズム機能部２１が、他のジョブステップを経由する経由ジョブステップをセットしたジョブステップにパイプ属性をセットし、該セットしたジョブステップの入力ファイルの数が複数と判定したジョブステップの属性をファイル処理に設定変更することによって、入力ファイルが複数のジョブステップのみをパイプ処理の属性に変更し、ファイル処理とパイプ処理の属性を付与したジョブ制御ファイルを生成する。またグループ化アルゴリズム機能部２２が、属性にパイプ処理が設定されたジョブステップに処理グループ番号を付し、該処理グループ番号が比較的大きいジョブステップに処理グループ番号が比較的小さいジョブを統合することによって、ジョブステップの実行順序を考慮したジョブ制御ファイルを生成する。
【選択図】図１



【特許請求の範囲】
【請求項１】
プロセスが実行する複数のジョブステップと該ジョブステップによりプロセスから入出力されるファイルとを記述した入力ジョブ制御ファイルを入力とし、該入出力ファイルをプロセスがパイプ処理を行うかファイル処理を行うかの属性を付与した属性付きジョブ制御ファイルを出力するジョブ管理サービスと、該ジョブ管理サービスから出力された属性付きジョブ制御ファイルのファイル属性を含むジョブステップを実行するジョブ実行制御部とを備えるジョブ管理システムであって、
該ジョブ管理サービス部が、
前記入力ジョブ制御ファイルを入力とし、該入力ジョブ制御ファイルに含まれるジョブステップの入力ファイルが経由する他のジョブステップを経由ジョブステップとしてセットする第１工程と、
該第１工程により経由ジョブステップをセットしたジョブステップにパイプ属性をセットする第２工程と、
該第２工程によりパイプ属性をセットしたジョブステップの入力ファイルの数が単数か否かを判定する第３工程と、
該第３工程により入力ファイルの数が単数でないと判定したジョブステップの属性をファイル処理に設定する第４工程と、
前記第３工程により入力ファイルの数が単数と判定したジョブステップの属性をパイプ処理に維持する設定を行う第５工程と、
前記ジョブ実行制御部が、
前記第４工程及び第５工程により設定したジョブステップを含む属性付きジョブ制御ファイルを入力とし、該入力した属性付きジョブ制御ファイルのファイル属性を含むジョブステップを実行する第６工程とを実行するジョブ管理システム。
【請求項２】
前記ジョブ管理システムが、ジョブステップを統合するグループ化アルゴリズム機能部を含み、該グループ化アルゴリズム機能部が、
前記第４工程に続き、
前記第４工程及び第５工程により設定した複数のジョブステップに処理グループ番号を付与する第７工程と、
該第７工程により処理グループ番号を付与したジョブステップに、該ジョブステップの入力ファイルを出力するジョブステップの内包ジョブステップ番号を付与する第８工程と、
該第８工程による内包ジョブステップ番号が共通するジョブステップを、同一の処理グループ番号の処理グループに統合する第９工程と、
該第９工程により統合した処理グループ番号のジョブステップの内包ジョブステップの入力ファイル属性が全てパイプ処理か否かを判定し、内包ジョブステップの入力ファイル属性が全てパイプ処理と判定したとき、該入力ファイル属性が全てパイプ処理と判定したジョブステップのみを統合する第１０工程と、
該第１０工程に統合したジョブステップと前記第５工程により設定したジョブステップを含む属性付きジョブ制御ファイルを前記ジョブ実行制御部に出力する第１１工程とを実行する請求項１記載のジョブ管理システム。
【請求項３】
前記グループ化アルゴリズム機能部が、前記第９工程により統合した入力ファイル属性が全てパイプ処理と判定したジョブステップの統合を行うとき、前記処理グループ番号の値が大きいジョブステップに、該処理グループ番号の値が大きいジョブステップに比べて処理グループ番号の値が小さいジョブステップを統合する第１２工程を実行する請求項２記載のジョブ管理システム。
【請求項４】
前記グループ化アルゴリズム機能部が、前記第１２工程により統合した処理グループ番

号の値が小さいジョブステップの処理グループ番号を削除する第１３工程を実行する請求項３記載のジョブ管理システム。
【請求項５】
前記ジョブ実行制御部が、前記属性付きジョブ制御ファイルのファイル属性の記述がないジョブステップをファイル処理と判定し、パイプ処理との属性が記述されているジョブステップをパイプ処理と判定して、ジョブステップを実行する請求項１から４何れかに載のジョブ管理システム。
【請求項６】
プロセスが実行する複数のジョブステップと該ジョブステップによりプロセスから入出力されるファイルとを記述した入力ジョブ制御ファイルを入力とし、該入出力ファイルをプロセスがパイプ処理を行うかファイル処理を行うかの属性を付与した属性付きジョブ制御ファイルを出力するジョブ管理サービスと、該ジョブ管理サービスから出力された属性付きジョブ制御ファイルのファイル属性を含むジョブステップを実行するジョブ実行制御部とを備えるジョブ管理システムのジョブ実行方法であって、
該ジョブ管理サービス部に、
前記入力ジョブ制御ファイルを入力とし、該入力ジョブ制御ファイルに含まれるジョブステップの入力ファイルが経由する他のジョブステップを経由ジョブステップとしてセットする第１工程と、
該第１工程により経由ジョブステップをセットしたジョブステップにパイプ属性をセットする第２工程と、
該第２工程によりパイプ属性をセットしたジョブステップの入力ファイルの数が単数か否かを判定する第３工程と、
該第３工程により入力ファイルの数が単数でないと判定したジョブステップの属性をファイル処理に設定する第４工程と、
前記第３工程により入力ファイルの数が単数と判定したジョブステップの属性をパイプ処理に維持する設定を行う第５工程とを実行させ、
前記ジョブ実行制御部に、
前記第４工程及び第５工程により設定したジョブステップを含む属性付きジョブ制御ファイルを入力とし、該入力した属性付きジョブ制御ファイルのファイル属性を含むジョブステップを実行する第６工程とを実行させるジョブ管理方法。
【請求項７】
前記ジョブ管理システムが、ジョブステップを統合するグループ化アルゴリズム機能部を含み、該グループ化アルゴリズム機能部に、
前記第４工程に続き、
前記第４工程及び第５工程により設定した複数のジョブステップに処理グループ番号を付与する第７工程と、
該第７工程により処理グループ番号を付与したジョブステップに、該ジョブステップに入力するファイルを出力するジョブステップの内包ジョブステップ番号を付与する第８工程と、
該第８工程による内包ジョブステップ番号が共通するジョブステップを、同一の処理グループ番号の処理グループに統合する第９工程と、
該第９工程により統合した処理グループ番号のジョブステップの内包ジョブステップの入力ファイル属性が全てパイプ処理か否かを判定し、内包ジョブステップの入力ファイル属性が全てパイプ処理と判定したとき、該入力ファイル属性が全てパイプ処理と判定したジョブステップのみを統合する第１０工程と、
該第１０工程に統合したジョブステップと前記第５工程により設定したジョブステップを含む属性付きジョブ制御ファイルを前記ジョブ実行制御部に出力する第１１工程とを実行させる請求項６記載のジョブ管理方法。
【請求項８】
前記グループ化アルゴリズム機能部に、前記第９工程により統合した入力ファイル属性

が全てパイプ処理と判定したジョブステップの統合を行うとき、前記処理グループ番号の値が大きいジョブステップに、該処理グループ番号の値が大きいジョブステップに比べて処理グループ番号の値が小さいジョブステップを統合する第１２工程を実行させる請求項７記載のジョブ管理方法。

【請求項９】

前記グループ化アルゴリズム機能部に、前記第１２工程により統合した処理グループ番号の値が小さいジョブステップの処理グループ番号を削除する第１３工程を実行させる請求項８記載のジョブ管理方法。

【請求項１０】

前記ジョブ実行制御部に、前記属性付きジョブ制御ファイルのファイル属性の記述がないジョブステップをファイル処理と判定し、パイプ処理との属性が記述されているジョブステップをパイプ処理と判定し、ジョブステップを実行させる請求項６から９何れかに記載のジョブ管理方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、コンピュータ処理におけるパイプを利用したジョブ制御を自動的に行うジョブ管理システム及びジョブ管理方法に係り、特にプロセス間のデータ引継ぎによる異常終了を防止したジョブ管理システム及びジョブ管理方法に関する。

【背景技術】

【０００２】

一般にメインフレームコンピュータにおいて行われるプログラムのバッチ処理は、ＪＣＬ（ジョブ制御言語）を用いてジョブステップ（ジョブを構成する最小単位となる処理プログラム）を組み合わせたジョブを構成しておき、オペレータがジョブを投入すると、メインフレームコンピュータが前記ＪＣＬを解釈し、ジョブを実行することによって行われる。

【０００３】

このジョブの実行は、前述したジョブステップを指定の順番（並列処理を含む）に起動することによって行われる。ジョブステップは、非同期並列処理を行うために、プロセス（ＯＳからプロセス空間であるメモリリソースの割り当てを受けて処理を実行しているプログラムのこと。通常、プロセスは互いに影響を与えないように動作する）として起動される。ジョブのフローでは、あるジョブステップの出力が別のジョブステップの入力になることがあるため、プロセス間でデータを引継ぐ必要があるが、その方式にはファイル処理を用いる場合と、パイプ処理を用いる場合があり、この指定はＪＣＬにて行う。尚、パイプ処理とは、プログラム同士を組み合わせて利用するための一手法であって、１つのプロセス空間内における、ある処理の出力を別の処理の入力とする処理のことを言い、ファイル処理とは、あるプロセスが記憶装置に一旦ファイルを記憶させることによって、他のプロセスからのアクセスを許す処理を意味する。

【０００４】

このパイプ処理は、例えば図１０に示す如く、ある出力プロセス２０１から出力されたデータをパイプ２０２のバッファ２０３の所定の領域（図１０の例では「ｎ＋３件目」）に書きだし、「ｎ件目」のデータを次の入力プロセス２０４が読み込む処理を意味し、次の特徴がある。

（１）データはＦＩＦＯ（Ｆｉｒｓｔ　Ｉｎ，　Ｆｉｒｓｔ　Ｏｕｔ）の取り出し方式であり、バッファ２０３に入った順番で取り出される。

（２）出力プロセス２０１から出力されたデータは、バッファ２０３に保持される。

（３）入力プロセス２０４に読みこまれたデータはバッファ２０３から消去される。

（４）バッファ２０３に空きがなくなると、出力プログラムはバッファ２０３に空きが出るまで処理が停止する。

【０００５】

このパイプ処理を用いたデータ引継ぎには次の利点がある。
（４）ディスクＩ／Ｏが発生しないために高速。この理由は、外部記憶装置とのファイルＩ／Ｏ（ディスクＩ／Ｏ）は、主記憶装置（メモリ）へのＩ／Ｏやソケット通信（特に同一マシン上でのプロセス間通信としてのソケット通信）に対して低速のためである。
（５）入出力の依存関係がある複数のプロセスは並列実行可能。
（６）ファイル処理を用いた方法では、外部記憶装置へのファイル出力が全て完了しないと、それを入力とするジョブステップの起動ができないため、逐次処理にせざるを得ない。これに対し、パイプ処理を用いた方法ではファイル単位ではなく、ファイルを構成するデータの単位であるレコード単位にデータの入出力を行い、出力プロセスがバッファに出力したレコードを次の入力プロセスが読み込んで処理を行うことで並列実行が可能になる。

【０００６】
この並列実行処理は、図１１に示す如く、２つのパイプ２２１と２２４とが有り、相互に入出力関係があるプロセス２２０とプロセス２２３とプロセス２２５とが同時に起動され、プロセス２２０の出力がパイプ２２１のバッファ２２２に書き込まれると、プロセス２２３がパイプ２２１のバッファ２２２の「ｎ件目」のデータを読み込み、このデータを処理してパイプ２２４のバッファ２２６に出力を書きみ、このプロセス２２３の出力がパイプ２２４に書き込まれると、プロセス２２５がパイプ２２４のバッファ２２６の「ｎ件目」のデータを読み込む様に動作し、これらの動作によってプロセスが並列実行される。

【０００７】
尚、前記パイプ処理に関する技術が記載された文献としては下記特許文献１が挙げられ、この特許文献１には、入力プロセスが複数の場合や中間データを保存する場合、処理結果であるデータを出力する第１プロセス及び該第１プロセスが出力したデータを引き継いで処理を行う第２プロセスのいずれからもアクセス可能なバッファを設け、第１プロセスから出力されたデータを前記バッファに順次書き込む工程と、該バッファに書き込まれたデータを前記第２プロセスに読み込ませる工程と、前記第１プロセスにより書き込んだデータがバッファから全て第２プロセスに読み込ませたとき、バッファ領域を開放する工程とを実行することが記載されている。
【特許文献１】特開平７－２６２０２４号公報

【発明の開示】
【発明が解決しようとする課題】

【０００８】
前述の図１１にて説明した並列実行処理は、パイプ処理におけるバッファサイズには制限があるため、出力するバッファが全て埋まった場合、後続ジョブステップに読み取られることでバッファが空くまで出力プロセスの処理が停止する可能性があった。

【０００９】
この不具合を具体的に説明する。まず、メインフレーム時代のＪＣＬによる制御では、データの引継ぎ方式をＪＣＬ上で指定している。Ｃ／Ｓ型システムで実現しているパイプ処理引継ぎ方式でも、パイプ化するファイル処理をユーザが指定しなければならないことは同様であり、ＪＣＬによるメインフレーム時代と本質的に同義である。この場合、ＪＣＬ作成者がパイプ処理を指定すべきかファイル処理を指定すべきかを判断し、誤りなく指定する必要がある。また、性能を最大限に上げるには、パイプ処理を指定できる個所を漏れなく指定する必要がある。これを実現するには、ＪＣＬ作成者はジョブステップ間の入出力関係を全て分析し、出力ジョブステップが停止しない個所を特定しなければならない。この作業は難しく、また面倒なものなので、実際にはパイプ処理の有効性を理解しているエンジニアでもパイプ処理指定を行わないか、確実性が保証されている個所にしか指定しないことが多い。

【００１０】
この自動化にあたっては次の２点の課題があり、従来技術においては何れの課題も考慮していないと言う不具合があった。

【００１１】
（ａ）処理停止を回避すること。すなわち、パイプ処理にしてはいけない個所をパイプ処理指定しないようにすること。ここでいう処理停止とは、１つのプロセスに対する複数の入力の何れかが、他の入力を待つことによって処理が停止してしまうことを指し、この停止処理を図１２を参照して説明する。
【００１２】
図１２は、複数のパイプ５５～５８とから構成されるシステムを示し、プロセス５１の出力データをパイプ５５又はパイプ５６を介してプロセス５２及びプロセス５３が受け取り、プロセス５２の出力データをプロセス５４がパイプ５７を介して受け取り、プロセス５３の出力をプロセス５４がパイプ５７を介して受け取る構成を示している。
【００１３】
この構成のシステムは、プロセス５１の出力がプロセス５２側へ集中した場合、プロセス５３は入力があるまで待ち状態となり、このためプロセス５４がプロセス５３からのデータ入力がないために待ち状態となり、パイプ５７からのデータを読み込まなくなる。従って、パイプ５７からデータがプロセス５４に読み込まれないため、パイプ５７のバッファ６３が全て埋まった段階で前段のプロセス５２は待ち状態になり、パイプ５５からの入力を読み込まず、パイプ５５からデータが読み込まれないため、パイプ５５のバッファ６１が全て埋まった段階でプロセス５１は待ち状態になる。こうして、全てのプロセスが待ち状態になり、処理停止となる可能性があった。
【００１４】
（ｂ）処理順序を最適化すること。ファイル処理とパイプ処理を混在させ、実行可能状態のジョブステップを全て即時実行した場合、入力ファイルが存在しないプロセスが存在し、異常終了してしまう可能性があり、この異常終了を図１３を参照して説明する。
【００１５】
図１３は、プロセス５１からの出力データ６１をプロセス５３に入力し、プロセス５２からの出力データ６２をプロセス５３に入力し、これら両データを入力したプロセス５３が処理を実行する構成を示している。
【００１６】
この構成のシステムは、プロセス５１とプロセス５２とプロセス５３とを同時実行してしまうと、プロセス５３の入力のうちデータ６１をプロセス５１の処理が終わるまでプロセス５３が読み込めないため、プロセス５３は入力ファイルの一部が存在しないプロセスとなってしまい、異常終了してしまう可能性があった。
【００１７】
前述のパイプ処理による異常終了を防止するためには、オペレータがファイル処理を行うファイル処理とパイプ処理を行うファイルとを予め解析して設定しておかなければならず、このファイル処理とパイプ処理を大量の処理過程にわたって解析及び設定する作業がオペレータの多大な作業負担になると言う不具合があった。
【００１８】
これを具体的に説明すると、ＪＣＬ（ジョブ制御言語）を用いてジョブステップを組み合わせたジョブが、図１４（ａ）に示したジョブ制御ファイル１０の如く、実行プログラム名と、実行するジョブステップ名と、該ジョブステップに入力する入力ファイル情報と、該ジョブステップから出力される出力ファイル情報との各情報を格納した形式で表され、例えば、このジョブにより形成される処理ツリーが、図１４（ｂ）に示す如く、ｊｏｂｓｔｅｐ１（符号５１）がｆｉｌｅ６３及び６４を出力し、ｊｏｂｓｔｅｐ２（符号６５）が前記ｆｉｌｅ１を入力してｆｉｌｅ３（符号６７）を出力し、ｊｏｂｓｔｅｐ３（符号６６）が前記ｆｉｌｅ２１を入力してｆｉｌｅ４（符号６８）を出力するであり、ｆｉｌｅ２がパイプ処理すべきデータであり、ファイル記述のデフォルトがファイル処理の場合、図１４（ａ）の「出力ファイル２」の記述に示す如く、ｆｉｌｅ２の属性として「ｐｉｐｅ＝ｏｎ」の記述を加入しなければならず、多量のファイルを取り扱うジョブにおいては前述したオペレータによるファイル処理とパイプ処理との解析並びに設定が煩雑であ

り、これら作業がオペレータの多大な作業負担になると言う不具合があった。

【００１９】

本発明は、プロセス間のデータ引継ぎ方式の指定を自動化かつ最適化し、ＪＣＬ作成者の工数削減を実現するジョブ管理システム及びジョブ管理方法を提供することである。

【課題を解決するための手段】

【００２０】

前記目的を達成するために本発明は、プロセスが実行する複数のジョブステップと該ジョブステップによりプロセスから入出力されるファイルとを記述した入力ジョブ制御ファイルを入力とし、該入出力ファイルをプロセスがパイプ処理を行うかファイル処理を行うかの属性を付与した属性付きジョブ制御ファイルを出力するジョブ管理サービスと、該ジョブ管理サービスから出力された属性付きジョブ制御ファイルのファイル属性を含むジョブステップを実行するジョブ実行制御部とを備えるジョブ管理システムであって、

該ジョブ管理サービス部が、

前記入力ジョブ制御ファイルを入力とし、該入力ジョブ制御ファイルに含まれるジョブステップの入力ファイルが経由する他のジョブステップを経由ジョブステップとしてセットする第１工程と、

該第１工程により経由ジョブステップをセットしたジョブステップにパイプ属性をセットする第２工程と、

該第２工程によりパイプ属性をセットしたジョブステップの入力ファイルの数が単数か否かを判定する第３工程と、

該第３工程により入力ファイルの数が単数でないと判定したジョブステップの属性をファイル処理に設定する第４工程と、

前記第３工程により入力ファイルの数が単数と判定したジョブステップの属性をパイプ処理に維持する設定を行う第５工程と、

前記ジョブ実行制御部が、

前記第４工程及び第５工程により設定したジョブステップを含む属性付きジョブ制御ファイルを入力とし、該入力した属性付きジョブ制御ファイルのファイル属性を含むジョブステップを実行する第６工程とを実行することを第１の特徴とする。

【００２１】

また本発明は、該第１の特徴のジョブ管理システムにおいて、前記ジョブ管理システムが、ジョブステップを統合するグループ化アルゴリズム機能部を含み、該グループ化アルゴリズム機能部が、

前記第４工程に続き、

前記第４工程及び第５工程により設定した複数のジョブステップに処理グループ番号を付与する第７工程と、

該第７工程により処理グループ番号を付与したジョブステップに、該ジョブステップに入力するファイルを出力するジョブステップの内包ジョブステップ番号を付与する第８工程と、

該第８工程による内包ジョブステップ番号が共通するジョブステップを、同一の処理グループ番号の処理グループに統合する第９工程と、

該第９工程により統合した処理グループ番号のジョブステップの内包ジョブステップの入力ファイル属性が全てパイプ処理か否かを判定し、内包ジョブステップの入力ファイル属性が全てパイプ処理と判定したとき、該入力ファイル属性が全てパイプ処理と判定したジョブステップのみを統合する第１０工程と、

該第１０工程に統合したジョブステップと前記第５工程により設定したジョブステップを含む属性付きジョブ制御ファイルを前記ジョブ実行制御部に出力する第１１工程とを実行することを第２の特徴とする。

【００２２】

また本発明は、該第２の特徴のジョブ管理システムにおいて、前記グループ化アルゴリズム機能部が、前記第９工程により統合した入力ファイル属性が全てパイプ処理と判定し

たジョブステップの統合を行うとき、前記処理グループ番号の値が大きいジョブステップに、該処理グループ番号の値が大きいジョブステップに比べて処理グループ番号の値が小さいジョブステップを統合する第１２工程を実行することを第３の特徴とする。

【００２３】

また本発明は、前記第３の特徴のジョブ管理システムにおいて、前記グループ化アルゴリズム機能部が、前記第１２工程により統合した処理グループ番号の値が小さいジョブステップの処理グループ番号を削除する第１３工程を実行することを第４の特徴とする。

【００２４】

また本発明は、前記何れかの特徴のジョブ管理システムにおいて、前記ジョブ実行制御部が、前記属性付きジョブ制御ファイルのファイル属性の記述がないジョブステップをファイル処理と判定し、パイプ処理との属性が記述されているジョブステップをパイプ処理と判定して、ジョブステップを実行することを第５の特徴とする。

【００２５】

更に本発明は、プロセスが実行する複数のジョブステップと該ジョブステップによりプロセスから入出力されるファイルとを記述した入力ジョブ制御ファイルを入力とし、該入出力ファイルをプロセスがパイプ処理を行うかファイル処理を行うかの属性を付与した属性付きジョブ制御ファイルを出力するジョブ管理サービスと、該ジョブ管理サービスから出力された属性付きジョブ制御ファイルのファイル属性を含むジョブステップを実行するジョブ実行制御部とを備えるジョブ管理システムのジョブ実行方法であって、

該ジョブ管理サービス部に、

前記入力ジョブ制御ファイルを入力とし、該入力ジョブ制御ファイルに含まれるジョブステップの入力ファイルが経由する他のジョブステップを経由ジョブステップとしてセットする第１工程と、

該第１工程により経由ジョブステップをセットしたジョブステップにパイプ属性をセットする第２工程と、

該第２工程によりパイプ属性をセットしたジョブステップの入力ファイルの数が単数か否かを判定する第３工程と、

該第３工程により入力ファイルの数が単数でないと判定したジョブステップの属性をファイル処理に設定する第４工程と、

前記第３工程により入力ファイルの数が単数と判定したジョブステップの属性をパイプ処理に維持する設定を行う第５工程とを実行させ、

前記ジョブ実行制御部に、

前記第４工程及び第５工程により設定したジョブステップを含む属性付きジョブ制御ファイルを入力とし、該入力した属性付きジョブ制御ファイルのファイル属性を含むジョブステップを実行する第６工程とを実行させることを第６の特徴とする。

【００２６】

また本発明は、該第６の特徴のジョブ管理方法において、前記ジョブ管理システムが、ジョブステップを統合するグループ化アルゴリズム機能部を含み、該グループ化アルゴリズム機能部に、

前記第４工程に続き、

前記第４工程及び第５工程により設定した複数のジョブステップに処理グループ番号を付与する第７工程と、

該第７工程により処理グループ番号を付与したジョブステップに、該ジョブステップに入力するファイルを出力するジョブステップの内包ジョブステップ番号を付与する第８工程と、

該第８工程による内包ジョブステップ番号が共通するジョブステップを、同一の処理グループ番号の処理グループに統合する第９工程と、

該第９工程により統合した処理グループ番号のジョブステップの内包ジョブステップの入力ファイル属性が全てパイプ処理か否かを判定し、内包ジョブステップの入力ファイル属性が全てパイプ処理と判定したとき、該入力ファイル属性が全てパイプ処理と判定した

ジョブステップのみを統合する第１０工程と、
該第１０工程に統合したジョブステップと前記第５工程により設定したジョブステップを含む属性付きジョブ制御ファイルを前記ジョブ実行制御部に出力する第１１工程とを実行させることを第７の特徴とする。
【００２７】
また本発明は、前記第７の特徴のジョブ管理方法において、前記グループ化アルゴリズム機能部に、前記第９工程により統合した入力ファイル属性が全てパイプ処理と判定したジョブステップの統合を行うとき、前記処理グループ番号の値が大きいジョブステップに、該処理グループ番号の値が大きいジョブステップに比べて処理グループ番号の値が小さいジョブステップを統合する第１２工程を実行させることを第８の特徴とする。
【００２８】
また本発明は、前記第８の特徴のジョブ管理方法において、前記グループ化アルゴリズム機能部に、前記第１２工程により統合した処理グループ番号の値が小さいジョブステップの処理グループ番号を削除する第１３工程を実行させることを第９の特徴とする。
【００２９】
また本発明は、前記第６から第９何れかの特徴のジョブ管理方法において、前記ジョブ実行制御部に、前記属性付きジョブ制御ファイルのファイル属性の記述がないジョブステップをファイル処理と判定し、パイプ処理との属性が記述されているジョブステップをパイプ処理と判定し、ジョブステップを実行させることを第１０の特徴とする。
【発明の効果】
【００３０】
本発明によるジョブ管理システム及びジョブ管理方法は、ジョブ管理サービス部が、入力ファイルが他のジョブステップを経由する経由ジョブステップをセットしたジョブステップにパイプ属性をセットし、該経由ジョブステップをセットしたジョブステップの入力ファイルの数が単数でないと判定したジョブステップの属性をファイル処理に設定変更することによって、入力ファイルが複数ジョブステップのみをパイプ処理の属性に変更し、ファイル処理とパイプ処理の属性を付与した属性付きジョブ制御ファイルを生成し、ジョブを停止処理が発生することなく実行することができる。また属性にパイプ処理が設定されたジョブステップに処理グループ番号を付し、該処理グループ番号が比較的大きいジョブステップに処理グループ番号が比較的小さいジョブを統合することによって、実行順序を適正化してプロセス間のデータ引継ぎによる異常終了を防止した属性付きジョブ制御ファイルを生成し、実行することができる。
【発明を実施するための最良の形態】
【００３１】
以下、本発明によるジョブ管理システム及びジョブ管理方法の一実施形態を図面を参照して詳細に説明する。図１は、本発明の一実施形態によるジョブ管理方法が適用されるジョブ管理システムの全体構成を示す図、図２は本発明の第１の本実施形態によるジョブ制御ファイルを示す図、図３本実施形態による処理ツリーを示す図、図４は本実施形態によるファイル化アルゴリズムのフローチャート図、図５は第２の実施形態によるジョブ制御ファイルを示す図、図６は本実施形態による処理ツリーを示す図、図７は本実施形態によるグループ化アルゴリズムを示す図、図８は本実施形態によるグループ化完了時の状態を説明するための図、図９は本実施形態によるグループ統合完了時の状態を説明するための図である。
【００３２】
［構成］
本発明の一実施形態によるジョブ管理システム及びジョブ管理方法を実現するコンピュータシステムは、図１に示す如く、ジョブ制御ファイル１０を読み込み、本実施形態の特徴である処理停止状態を回避するジョブ管理方法を実行するジョブ管理サービス機能部２０と、該ジョブ管理サービス機能部２０により変更されたジョブデータを実行し、その実行結果であるジョブ４０を出力するジョブ実行制御部３０とから構成される。これらジョ

ブ管理サービス機能部２０及びジョブ実行制御部３０は、コンピュータのハードウェア又はソフトウェアによって構成される。

【００３３】

本発明の第１の実施形態の対象となるジョブ制御ファイル１０は，図２に示す如く，ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ１」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ１」、入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ０」、第１の出力ファイル「ＯＦＩＬＥ０１」として「ｆｉｌｅ１」、第２の出力ファイル「ＯＦＩＬＥ０２」として「ｆｉｌｅ２」が設定され、同様に、ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ２」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ２」、入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ１」、出力ファイル「ＯＦＩＬＥ０１」として「ｆｉｌｅ３」が設定され、ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ３」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ３」、入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ２」、出力ファイル「ＯＦＩＬＥ０１」として「ｆｉｌｅ４」が設定され、ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ４」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ４」、第１の入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ３」、第２の入力ファイル「ＩＦＩＬＥ０２」として「ｆｉｌｅ５」、出力ファイル「ＯＦＩＬＥ０１」として「ｆｉｌｅ５」を設定記述している。このように本実施形態の対象となるジョブ制御ファイル１０は、前述したパイプ処理かファイル処理かを意識せずにオペレータがプログラム使用のフローチャート等を基に作成したものである。

【００３４】

図２に示したジョブ制御ファイル１０の処理ツリーは、図３に示す如く、「ｆｉｌｅ０」（符号７０）を「ｊｏｂｓｔｅｐ１」（符号７１）が読み込んだとき、該「ｊｏｂｓｔｅｐ１」が、「ｆｉｌｅ１及び２」（符号７２，７３）を出力し、前記「ｆｉｌｅ１」を読み込んだ「ｊｏｂｓｔｅｐ２」（符号７４）が「ｆｉｌｅ３」（符号７６）を出力し、前記「ｆｉｌｅ２」を読み込んだ「ｊｏｂｓｔｅｐ３」（符号７５）が「ｆｉｌｅ４」（符号７７）を出力し、これら出力された「ｆｉｌｅ３及び４」（符号７６，７７）を読み込んだ「ｊｏｂｓｔｅｐ７８」（符号７８）が「ｆｉｌｅ５」（符号７９）を出力する様に構成される。本例においては、「ｆｉｌｅ１及び２」をファイル処理の対象とし、他の「ｆｉｌｅ」をパイプ処理の対象とする必要があるものとし、詳細は後述する。

【００３５】

また本発明の第２の実施形態の対象となるジョブ制御ファイル１０は，図５に示す如く，図２に示した定義ファイル同様に、ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ１」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ１」、入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ０」、出力ファイル「ＯＦＩＬＥ０１」として「ｆｉｌｅ１」が設定され、ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ２」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ２」、入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ１」、出力ファイル「ＯＦＩＬＥ０１」として「ｆｉｌｅ２」が設定され、ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ３」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ３」、入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ１」、第１の出力ファイル「ＯＦＩＬＥ０１」として「ｆｉｌｅ３－１」、第２の出力ファイル「ＯＦＩＬＥ０２」として「ｆｉｌｅ３－２」が設定され、ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ４」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ４」、入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ２」、出力ファイル「ＯＦＩＬＥ０１」として「ｆｉｌｅ４」が設定され、ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ５」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ５」、第１の入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ３－１」、第２の入力ファイル「ＩＦＩＬＥ０２」として「ｆｉｌｅ３－２」が設定され、ジョブステップ名「ＪＯＢＳＴＥＰ」の「ｊｏｂｓｔｅｐ６」のジョブが、プログラム名「ＰＧＭ」として「ＰＧ６」、第１の入力ファイル「ＩＦＩＬＥ０１」として「ｆｉｌｅ４」、第２の入力ファイル「ＩＦＩＬＥ０２」と

して「ｆｉｌｅ５」、出力ファイル「ＯＦＩＬＥ０１」として「ｆｉｌｅ６」を設定記述している。

【００３６】

図５に示したジョブ制御ファイル１０の処理ツリーは、図６に示す如く「ｆｉｌｅ１２０」（符号１２０）を読み込んだ「ｊｏｂｓｔｅｐ１」（符号１２１）が「ｆｉｌｅ１」（符号１２２）を出力し、該「ｆｉｌｅ１」を読み込んだ「ｊｏｂｓｔｅｐ２」（符号１２３）が「ｆｉｌｅ２」（符号１２５）を出力し、前記「ｆｉｌｅ１」を読み込んだ「ｊｏｂｓｔｅｐ３」（符号１２４）が「ｆｉｌｅ３－１」（符号１２６）及び「ｆｉｌｅ３－２」（符号１２７）を出力し、「ｊｏｂｓｔｅｐ４」（符号１２８）が前記「ｆｉｌｅ２」を読み込んで「ｆｉｌｅ４」（符号１３０）を出力し、「ｊｏｂｓｔｅｐ５」（符号１２９）が前記「ｆｉｌｅ３－１及び３－２」を読み込んで「ｆｉｌｅ５」（符号１３１）を出力し、前記「ｆｉｌｅ４及び５」を読み込んだ「ｊｏｂｓｔｅｐ６」（符号１３２）が「ｆｉｌｅ６」（符号１３３）を出力する様に構成される。本例においては、「ｆｉｌｅ１」と「ｆｉｌｅ３－１」と「ｆｉｌｅ３－２」をファイル処理の対象とし、他の「ｆｉｌｅ」をパイプ処理の対象とする必要があるものとし、詳細は後述する。

【００３７】

［動作］

本実施形態によるジョブ管理サービス機能部２０は、ジョブが投入されると前述したジョブ制御ファイル１０の記述内容を読み込み、各ジョブとファイルデータとの依存関係を解析し、パイプ処理による不具合が生じるジョブステップのパイプ処理を行うための属性（例えば「ｐｉｐｅ＝ｏｎ」）を付与したジョブ制御ファイルを後述するファイル化アルゴリズム及びグループ化アルゴリズムを用いて自動的に生成するものであって、まず、前述した図２に示したジョブ制御ファイル１０の変換処理について説明する。

【００３８】

［ファイル化アルゴリズム］

ジョブ管理サービス機能部２０は、図２に示したジョブ制御ファイル１０を読み込み、ファイル化アルゴリズム機能部２１が、図４に示したファイル化アルゴリズムを用いて解析する。この解析アルゴリズムは、前述した図１２にて説明した処理停止状態が発生しないことを保証するためには、（ａ）全てのジョブステップの入力が単数であること、又は（ｂ）複数の入力全てが互いに同一のジョブステップをパイプによって経由していないことの何れか一方の条件を満たせば良いこと、即ち「ジョブステップと入出力ファイルの関係が閉領域を構成するときに処理停止が発生する可能性があること」に着目したものである。以下、図４に示したアルゴリズムを「ファイル化アルゴリズム」と呼ぶ。

【００３９】

まず、図４では、ファイル数をＮ（＞＝１）と表記し、各入出力ファイルをＦＩＬＥ（ｉ）（ｉ＝１，…，Ｎ）と表記し、ジョブステップ数をＭ（＞＝１）と表記し、各ジョブステップをＪＯＢＳＴＥＰ（ｉ）（ｉ＝１，…，Ｍ）と表記しており、以下、本アルゴリズムを順を追って説明する。

【００４０】

本ファイル化アルゴリズムによる処理は、入出力ファイルのＦＩＬＥ（ｉ）の変数「ｉ」に値「１」を代入するステップＳ５０２と、該ＦＩＬＥ（ｉ）に経由ジョブステップをセットするステップＳ５０４と、変数「ｉ」が値「Ｎ」に達するまで該ステップＳ５０４を繰り返すステップＳ５０３及び５０５とを実行することにより、全ての入出力ファイルに対して経由ジョブステップをセットする。

【００４１】

この経由ジョブステップとは、そのファイルが作成されるまでに経由してきたジョブステップのことであり、例えば、任意のファイルＦＩＬＥ（ｎ）が作成されるまでにＪＯＢＳＴＥＰ（ａ）→（中間ファイル）→ＪＯＢＳＴＥＰ（ｂ）→（中間ファイル）→ＪＯＢＳＴＥＰ（ｃ）→ＦＩＬＥ（ｎ）出力とジョブステップが実行されている場合、ＦＩＬＥ（ｎ）の経由ジョブステップはＪＯＢＳＴＥＰ（ａ）とＪＯＢＳＴＥＰ（ｂ）とＪＯＢＳ

ＴＥＰ（ｃ）となる。

【００４２】

次いで本処理は、変数「ｉ」に値「１」を代入するステップＳ５０６と、ＪＯＢＳＴＥＰ（ｉ）の出力をパイプに設定するステップＳ５０８と、該ステップＳ５０８を変数「ｉ」が値「Ｍ」に達するまで繰り返すステップＳ５０７及び５０９とを実行することによって、全てのジョブステップに対して一旦出力するファイルの属性をパイプと設定する。

【００４３】

即ち、本ファイル化アルゴリズムは、まず、ステップ５０４により全ての入出力ファイルに対して経由ジョブステップをセットすると共に、ステップ５０８により全てのジョブステップの出力ファイルの属性を一旦パイプ処理として設定する処理を実行する。具体的には、ジョブ制御ファイルの該当出力ファイルに「ｐｉｐｅ＝ｏｎ」の記述を加える処理を行う。

【００４４】

更に本処理は、変数「ｉ」に値「１」を代入するステップＳ５１１と、ＪＯＢＳＴＥＰ（ｉ）の入力ファイル数Ｓが１より大きいか否か（即ち、入力ファイルが単数か複数か）を判定し、大きくないと判定したとき（単数と判定した）に後述するステップＳ５２８に移行するステップＳ５１２を実行する。このステップＳ５１２による判定は、前述した通り入力が単数の場合は片方の入力待ちによる処理停止が発生し得ない原理を利用し、これを判定するためである。

【００４５】

次いで本処理は、前記ステップＳ５１２において入力ファイルが複数と判定したとき、変数「ｊ」に値「１」を代入するステップＳ５２０と、変数「ｋ」に値「ｊ＋１」を代入するステップＳ５２２と、ＦＩＬＥ（ＪＯＢＳＴＥＰ（ｉ），ｊ）とＦＩＬＥ（ＪＯＢＳＴＥＰ（ｉ），ｋ）の経由ジョブステップに一致がないか否かを判定するステップＳ５２４を実行する。このステップＳ５２４は、入力ファイルの間で経由ジョブステップに共通のものがあるか判定するものであって、共通のものがあると判定したとき、図１３で説明した処理停止が発生する可能性があるため、出力ファイルの属性をファイルに指定するステップＳ５２５を実行する。このステップＳ５２５は、例えば、ＪＯＢＳＴＥＰ（ｎ）の入力ファイルがＦＩＬＥ（ａ）とＦＩＬＥ（ｂ）であり、ＦＩＬＥ（ａ）の経由ジョブステップがＪＯＢＳＴＥＰ（ｓ）とＪＯＢＳＴＥＰ（ｔ）、ＦＩＬＥ（ｂ）の経由ジョブステップがＪＯＢＳＴＥＰ（ｓ）とＪＯＢＳＴＥＰ（ｕ）である場合、ＪＯＢＳＴＥＰ（ｓ）が共通しているため、ＪＯＢＳＴＥＰ（ｓ）の出力をファイルに指定することによって、閉領域を崩して処理停止を防止する。

【００４６】

即ち、本ファイル化アルゴリズムは、ステップＳ５１２において、ジョブステップの入力ファイル数を判定し、入力ファイル数が単数と判定したとき、入力が単数の際には片方入力待ちによる停止処理が発生することがないため、パイプ処理属性とすることを確定（ステップＳ５０８により入出力ファイルに付与した属性「ｐｉｐｅ＝ｏｎ」を残したままとすること）し、入力ファイル数が単数でないと判定したとき、入力ジョブステップの間で経由ジョブステップに共通のものがあるか判定し、共通の経由ジョブステップがあると判定したとき、処理停止の可能性があるため、出力ファイルの属性をファイルとする。具体的には、ジョブ制御ファイルの入出力ファイルにステップ５０８で付与した「ｐｉｐｅ＝ｏｎ」の記述を削除する処理である。

【００４７】

従って、本実施形態によるファイル化アルゴリズムは、入力するファイル数が単数のジョブステップをファイル化するようにジョブ制御ファイルを書き換えることによって、停止処理の発生を防止したジョブ制御ファイルを図１５に示す如く、生成することができる。図１５の例では、図３に示した処理ツリーを参照すれば明らかな如く、ｆｉｌｅ１及び２（符号７２及び７３）を除く入出力ファイルに属性「ｐｉｐｅ＝ｏｎ」を付与した形に生成することができる。尚、最初と最後のｆｉｌｅ０及び５は、入力基及出力データのた

め属性はファイルである。

【００４８】

前述の実施形態によるファイル化アルゴリズムにより生成したジョブ制御ファイルは、ファイル処理とパイプ処理とが混在し、実行可能なジョブステップを同時に起動させた際には、入力ファイルが存在しないプロセスが存在する可能性があるため、ジョブステップを実行する順番を最適化する必要があり、この最適化処理を行うグループ化アルゴリズムを次に説明する。

【００４９】

［グループ化アルゴリズム］

本実施形態によるジョブ管理サービス機能部２０は、第２実施形態である図５に示したジョブ制御ファイル１０を読み込み、ファイル化アルゴリズム機能部２１が前述の図４に示したファイル化を実行した後、グループ化アルゴリズム機能部２２が、パイプ及びファイルをグループ化する処理を実行するものであって、この処理を図７を参照して説明する。以下、図７のアルゴリズムを「グループ化アルゴリズム」と呼び、図７の例では、ジョブステップ数をＭ（＞＝１）とし、各ジョブステップをＪＯＢＳＴＥＰ（ｉ）（ｉ＝１，…，Ｍ）と表記する。また、ＧＲ（ｎ）は処理グループを意味する。この処理グループとは、パイプによって同時に実行するジョブステップの集まりのことである。

【００５０】

［グループ化処理］

さて、このグループ化アルゴリズムは、図７に示す如く、変数「ｉ」及び「ｊ」に値「１」を代入するステップＳ８０１と、変数「ｉ」が値「１」かを判定することによって最初のジョブステップか否かを判定するステップＳ８０３と、該ステップＳ８０３により最初のジョブステップと判定したときに処理グループＧＲ（ｉ）を作成するステップＳ８０６と、該ステップＳ８０６に続いて変数「ｊ」に値「１」を加算してカウントアップするステップＳ８０８と、前記ステップＳ８０３において変数「ｉ」が値「１」でない（最初のジョブステップでない）と判定したとき、入力ファイルに「ファイル」が含まれるか否かを判定し、有ると判定したときに前記ステップＳ８０６に移行するステップＳ８０４と、該ステップＳ８０４において「ファイル」が含まれていないと判定したとき、入力ファイルが複数か否かを判定し、複数と判定したときに前記ステップＳ８０６に移行するステップＳ８０５と、該ステップＳ８０５において複数でないと判定したとき、変数「ｉ」に値「１」を加算してカウントアップするステップＳ８０９と、該ステップＳ８０９と変数「ｉ」が値「Ｍ」と等しくなるまで繰り返すステップＳ８０２とを実行する。

【００５１】

これらステップＳ８０２～８０９間の処理によって、本グループ化アルゴリズムは、（１）最初のジョブステップか、（２）入力ファイルの属性はファイルか、（３）入力ファイルが複数あるかの判定を行い、何れか１つの条件に当てはまる場合、新規に処理グループＧＲ（ｉ）をステップＳ８０６により作成し、作成した処理グループの内包ジョブステップ（その処理グループに含まれるジョブステップ）に対象のジョブステップをステップＳ８０７により追加するように処理を行う。

【００５２】

次いで本グループ化アルゴリズムは、ジョブステップの変数「Ｎ」から値「ｊ－１」を代入するステップＳ８１０と、変数「ｉ」に「１」を代入するステップＳ８１１と、ｊＯＢＳＴＥＰ（ｉ）が他の何れかの処理グループに内包されているか否かを判定するステップＳ８１３と、該ステップＳ８１３において内包されていないと判定したとき、ＪＯＢＳＴＥＰ（ｉ）を、入力マスタを出力したジョブステップが属する処理グループに追加するステップＳ８２０と、前記ステップＳ８１３又は８２０に続き、変数「ｉ」に値「１」を加算してカウントアップするステップＳ８２１と、該ステップＳ８２１との間で変数「ｉ」が値「Ｍ」と等しくなるまで繰り返すステップＳ８１２とを実行する。

【００５３】

これらステップＳ８１２～８２１による処理によってグループ化アルゴリズムは、全て

のジョブステップに対し、既に何れかの処理グループに含まれているかを判定する工程と、該工程において既に何れかの処理グループに含まれていないと判定したとき、入力ファイルを出力するジョブステップを内包する処理グループに、対象のジョブステップを内包ジョブステップとして追加する処理を行う。換言すれば、これらステップは、処理グループが複数存在するとき、後ろから（後から作成したものから）順に統合を行う。この統合は、対象となる処理グループの内包ジョブステップの入力が全てパイプの場合、内包ジョブステップの入力ファイルを出力するジョブステップが属する処理グループ（依存処理グループと呼ぶ）を対象となる処理グループに統合する。処理グループの統合は、統合元の処理グループの内包ジョブステップを全て統合先の内包ジョブステップに追加した後、統合元処理グループを削除することで行う。入力が全てパイプのもののみを統合することにより、入出力の矛盾は回避されており、後ろから統合することで処理順の整合性は保持されている。

【００５４】

この結果、本処理は、あるジョブステップが入力するファイルを出力するジョブステップは必ず１つとなるため、漏れも重複も存在せず、全てのジョブステップが何らかの処理グループに属させることができる。この手順で作成された処理グループは、それ自体は確実に入出力の矛盾が発生しない単位となっているため、作成順に処理を実行しても矛盾は発生しないが、最適化が不十分（同時に実行可能なジョブステップが同時に実行されない可能性がある）なことが想定される。このため本グループ化アルゴリズムにおいては、次の処理グループの統合処理を行う。

【００５５】

［処理グループ統合処理］

次いで本実施形態によるグループ化アルゴリズムは、前記ステップＳ８２２に続き、処理グループＧＲ（ｉ）が存在するか否かの判定を行い、存在しないと判定したときに後述するステップＳ８２７に移行するステップＳ８２４と、該ステップＳ８２４において処理グループＧＲ（ｉ）が存在すると判定したとき、存在を確認した処理グループＧＲ（ｉ）に内包されているジョブステップの入力ファイルが「パイプ」のみか否かの判定を行い、「パイプ」のみでないと判定したときにステップＳ８２７に移行するステップＳ８２８と、該ステップＳ８２８において、入力ファイルが「パイプ」のみと判定したとき、処理グループＧＲ（ｉ）に、処理グループＧＲ（ｉ）への入力マスタを出力する処理グループＧＲ（ｘ）に属するジョブステップを全て追加するステップＳ８２５と、該処理グループＧＲ（ｘ）を削除するステップＳ８２６と、変数「ｉ」に値「ｉ－１」を代入してカウントダウンするステップＳ８２７と、該ステップＳ８２７迄の処理を変数「ｉ」が値「２」に達するまで繰り返すステップＳ８２３とを実行する。

【００５６】

これらステップＳ８２３～８２８による処理による処理グループ統合処理は、処理グループが２つ以上ある場合に、後ろから（後から作成したものから）順に統合を行い、対象となる処理グループの内包ジョブステップの入力が全てパイプの場合、内包ジョブステップの入力ファイルを出力するジョブステップが属する処理グループ（依存処理グループと呼ぶ）を対象となる処理グループに統合する処理を行う。この処理グループの統合は、統合元の処理グループの内包ジョブステップを全て統合先の内包ジョブステップに追加した後、統合元処理グループを削除することにより行う。入力が全てパイプのもののみ統合することによって、入出力の矛盾は回避されており、後ろから統合することで処理順の整合性は保持される。

【００５７】

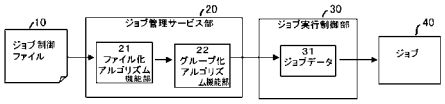
このグループ化アルゴリズムの処理を図６に示す処理ツリーを用いて説明すると、本実施形態によるグループ化アルゴリズムは、次（１）～（３）の処理を行う。

（１）「ｊｏｂｓｔｅｐ１」（符号１２０）と、入力がファイルである「ｊｏｂｓｔｅｐ２及びｊｏｂｓｔｅｐ３」（符号１２３及び１２４）と、複数入力が存在する「ｊｏｂｓｔｅｐ５及びｊｏｂｓｔｅｐ６」（符号１２９及び１３２）にて新規な処理グループを作

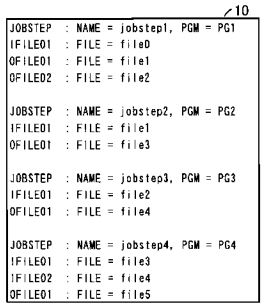
成する。
（２）処理グループに含まれていない「ｊｏｂｓｔｅｐ４」（符号１２８）を既存の処理グループに追加する。追加先は、入力ファイルを出力した「ｊｏｂｓｔｅｐ２」（符号１２５）の属する処理グループである。この段階での処理グループと各処理グループの内包ジョブステップ、及び依存処理グループの関係を図８に示す。
【００５８】
この図８に示した依存処理グループの関係は、符号（ａ）で示す処理グループ番号１が、依存処理グループがなく且つ内包ＪＯＢＳＴＥＰが「ｊｏｂｓｔｅｐ１」であり、符号（ｂ）で示す処理グループ番号２が、依存処理グループが「１」であり且つ内包ＪＯＢＳＴＥＰが「ｊｏｂｓｔｅｐ２」及び「ｊｏｂｓｔｅｐ４」であり、符号（ｃ）で示す処理グループ番号３が、依存処理グループが「１」であり且つ内包ＪＯＢＳＴＥＰが「ｊｏｂｓｔｅｐ３」であり、符号（ｄ）で示す処理グループ番号４が、依存処理グループが「３」であり且つ内包ＪＯＢＳＴＥＰが「ｊｏｂｓｔｅｐ５」であり、符号（ｅ）で示す処理グループ番号５が、依存処理グループが「２」及び「４」であり且つ内包ＪＯＢＳＴＥＰが「ｊｏｂｓｔｅｐ６」であることを表している。
（３）処理グループの統合（最適化）を行う。入力がパイプのみの処理グループには、図８の処理グループ番号５の処理グループ（内包ジョブステップはｓｔｅｐ６）が存在する。よって、処理グループ５の依存処理グループである処理グループ２及び処理グループ４を処理グループ５に統合し、処理グループ２と処理グループ４は削除することによって、処理グループの統合は完了する。
【００５９】
本実施形態によるジョブ管理システムは、図１に示したジョブ実行制御部３０が、前述した処理グループ番号の小さい順にジョブステップを実行することによって、停止処理がなく、且つ高速なバッチ処理を実行することができる。これを具体的に説明すると、本実施形態によるジョブ実行制御部３０が、最初に、図９に示した処理グループ１を実行し、「ｆｉｌｅ１」（符号１２２）を作成し、次に処理グループ３を実行し、「ｆｉｌｉｅ３－１及びｆｉｌｅ３－２」（符号１２６及び１２７）を作成し、最後に処理グループ５を実行することによって、停止処理がなく、且つ高速なバッチ処理を実行することができる。
【００６０】
このように本実施形態によるジョブ管理システム及びジョブ管理方法は、パイプ処理をファイル化する処理と、該ファイルを処理グループ化する処理と、該処理グループを統合化する処理と、ジョブイテップの実行順序を決める処理とを実行することによって、ジョブステップと入出力ファイルの関係が閉領域を構成しないようにファイル化アルゴリズムを実行させ、プログラムの停止を防止することができる。
【図面の簡単な説明】
【００６１】
【図１】本発明の一実施形態によるジョブ管理方法が適用されるジョブ管理システムの全体構成を示す図。
【図２】本発明の第１の本実施形態によるジョブ制御ファイルを示す図。
【図３】本実施形態による処理ツリーを示す図。
【図４】本実施形態によるファイル化アルゴリズムのフローチャート図。
【図５】本発明の第２の実施形態によるジョブ制御ファイルを示す図。
【図６】本実施形態による処理ツリーを示す図。
【図７】本実施形態によるグループ化アルゴリズムを示す図。
【図８】本実施形態によるグループ化完了時の状態を説明するための図。
【図９】本実施形態によるグループ統合完了時の状態を説明するための図。
【図１０】本発明の対象となるパイプ処理を説明するための図。
【図１１】本発明の対象となる並行パイプ処理を説明するための図。
【図１２】本発明の対象となる並行パイプ処理を説明するための図。

【図１３】従来の不適切な処理順序による異常終了状態を説明するための図。
【図１４】本発明の対象となるジョブ制御ファイル及び処理ツリーを示す図。
【図１５】第１の実施形態によるジョブ制御ファイルを示す図。
【図１６】第２の実施形態によるジョブ制御ファイルを示す図。
【符号の説明】
【００６２】
１０：ジョブ制御ファイル、２０：ジョブ管理サービス機能部、２１：ファイル化アルゴリズム機能部、２２：グループ化アルゴリズム機能部、３０：ジョブ実行制御部、４０：ジョブ。

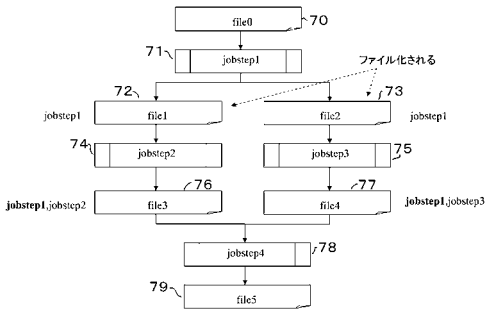
【図１】



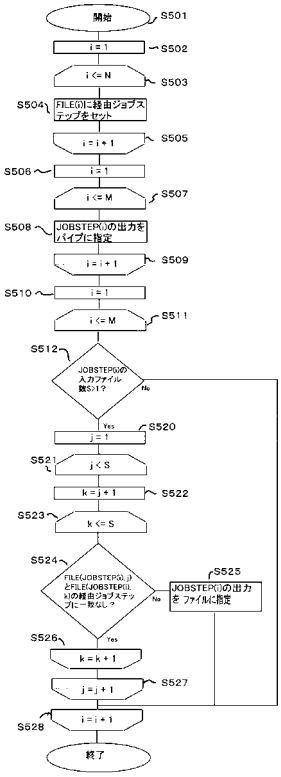
【図２】



【図３】



【図４】

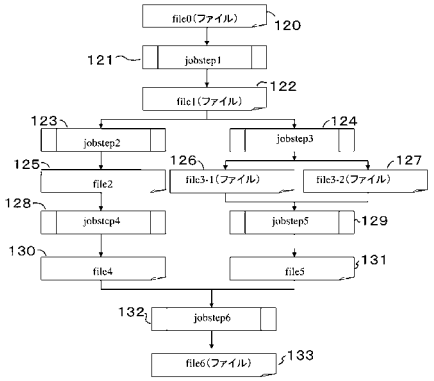


【図５】

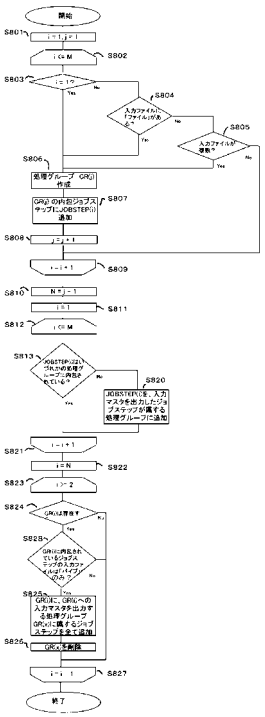
10

```
JOBSTEP  : NAME = jobstep1, PGM = PG1
IFILE01  : FILE = file0
OFILE01  : FILE = file1

JOBSTEP  : NAME = jobstep2, PGM = PG2
IFILE01  : FILE = file1
OFILE01  : FILE = file2

JOBSTEP  : NAME = jobstep3, PGM = PG3
IFILE01  : FILE = file1
OFILE01  : FILE = file3-1
OFILE01  : FILE = file3-2

JOBSTEP  : NAME = jobstep4, PGM = PG4
IFILE01  : FILE = file2
OFILE01  : FILE = file4

JOBSTEP  : NAME = jobstep5, PGM = PG5
IFILE01  : FILE = file3-1
IFILE02  : FILE = file3-2
OFILE01  : FILE = file5

JOBSTEP  : NAME = jobstep6, PGM = PG6
IFILE01  : FILE = file4
IFILE02  : FILE = file5
OFILE01  : FILE = file6
```

【図６】



【図７】



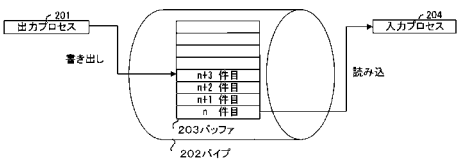
【図８】

(a)	処理グループ番号	1
	依存処理グループ	
	内包JOBSTEP	jobstep1
(b)	処理グループ番号	2
	依存処理グループ	1
	内包JOBSTEP	jobstep2, jobstep4
(c)	処理グループ番号	3
	依存処理グループ	1
	内包JOBSTEP	jobstep3
(d)	処理グループ番号	4
	依存処理グループ	3
	内包JOBSTEP	jobstep5
(e)	処理グループ番号	5
	依存処理グループ	2,4
	内包JOBSTEP	jobstep6

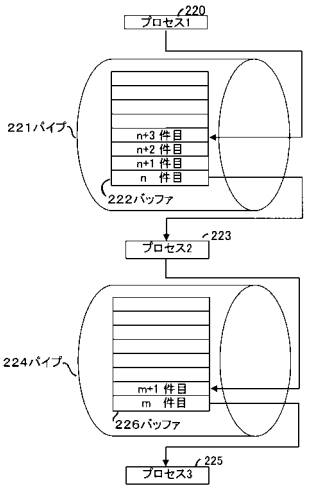
【図９】

(a)	処理グループ番号	1
	依存処理グループ	
	内包JOBSTEP	jobstep1
(b)	処理グループ番号	3
	依存処理グループ	1
	内包JOBSTEP	jobstep3
(c)	処理グループ番号	5
	依存処理グループ	1,3
	内包JOBSTEP	jobstep2, jobstep4, jobstep5,

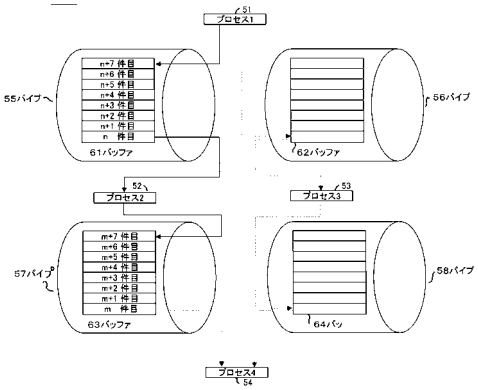
【図１０】



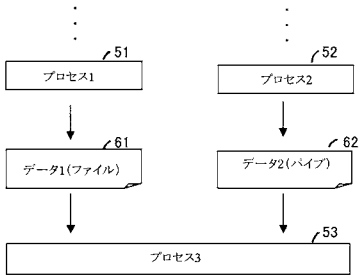
【図１１】



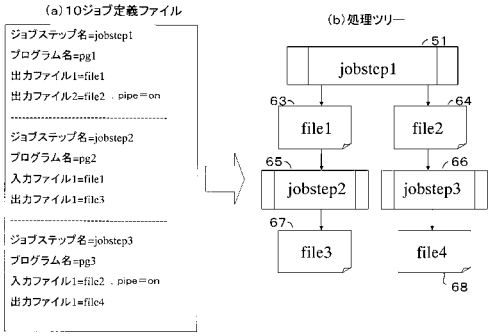
【図１２】



【図１３】



【図１４】



【図１５】

101

```
JOBSTEP  : NAME = jobstep1, PGM = PG1
IFILE01  : FILE = file0
OFILE01  : FILE = file1
OFILE02  : FILE = file2

JOBSTEP  : NAME = jobstep2, PGM = PG2
IFILE01  : FILE = file1
OFILE01  : FILE = file3, pipe=on

JOBSTEP  : NAME = jobstep3, PGM = PG3
IFILE01  : FILE = file2
OFILE01  : FILE = file4, pipe=on

JOBSTEP  : NAME = jobstep4, PGM = PG4
IFILE01  : FILE = file3, pipe=on
IFILE02  : FILE = file4, pipe=on
OFILE01  : FILE = file5
```

【図１６】

10

```
JOBSTEP  : NAME = jobstep1, PGM = PG1
IFILE01  : FILE = file0
OFILE01  : FILE = file1

JOBSTEP  : NAME = jobstep2, PGM = PG2
IFILE01  : FILE = file1
OFILE01  : FILE = file2, pipe=on

JOBSTEP  : NAME = jobstep3, PGM = PG3
IFILE01  : FILE = file1
OFILE01  : FILE = file3-1
OFILE01  : FILE = file3-2

JOBSTEP  : NAME = jobstep4, PGM = PG4
IFILE01  : FILE = file2, pipe=on
OFILE01  : FILE = file4, pipe=on

JOBSTEP  : NAME = jobstep5, PGM = PG5
IFILE01  : FILE = file3-1
IFILE02  : FILE = file3-2
OFILE01  : FILE = file5, pipe=on

JOBSTEP  : NAME = jobstep6, PGM = PG6
IFILE01  : FILE = file4, pipe=on
IFILE02  : FILE = file5, pipe=on
OFILE01  : FILE = file6
```