(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2004-30601
(P2004-30601A)
(43)公開日　平成16年1月29日(2004.1.29)

(51)Int.Cl.[7]		F I			テーマコード（参考）
G06F	9/445	G06F	9/06	640A	5B021
G06F	3/12	G06F	3/12	B	5B076
H04N	1/00	H04N	1/00	106B	5C062

審査請求 未請求　請求項の数 32　OL　（全 30 頁）

(21)出願番号　特願2003-120250(P2003-120250)
(22)出願日　平成15年4月24日(2003.4.24)
(31)優先権主張番号　特願2002-127079(P2002-127079)
(32)優先日　平成14年4月26日(2002.4.26)
(33)優先権主張国　日本国(JP)

(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号
(74)代理人　100070150
弁理士　伊東　忠彦
(72)発明者　田中　浩行
東京都大田区中馬込１丁目３番６号　株式会社リコー内
(72)発明者　大石　勉
東京都大田区中馬込１丁目３番６号　株式会社リコー内
(72)発明者　秋吉　邦洋
東京都大田区中馬込１丁目３番６号　株式会社リコー内

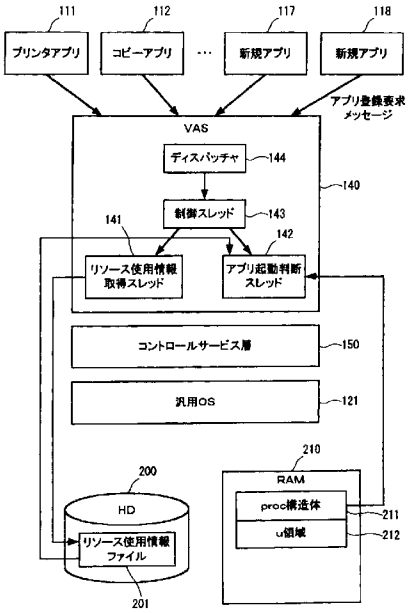
Fターム(参考)　5B021 AA01　AA19　BB01　CC06

最終頁に続く

(54)【発明の名称】リソース情報によりアプリケーション起動判断を行う装置及び方法

(57)【要約】
【課題】アプリケーション起動可否を判断し、複合機の安定性を図ること。
【解決手段】アプリケーションを実行させるためのリソースを備えた装置において、前記アプリケーションが起動される際に、前記装置におけるリソースの状態を示すリソース状態情報を取得し、取得したリソース状態情報と、前記アプリケーションが使用するリソースに関するリソース使用情報とに基づいて、前記アプリケーションに対して起動判断に関するメッセージを送信する手段を備えるように構成する。
【選択図】　図３

実施の形態１にかかる複合機のＶＡＳの構成と、ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの関係を示すブロック図である。



【特許請求の範囲】
【請求項１】
アプリケーションを実行させるためのリソースを備えた装置であって、
前記アプリケーションが起動される際に、前記装置におけるリソースの状態を示すリソース状態情報を取得し、取得したリソース状態情報と、前記アプリケーションが使用するリソースに関するリソース使用情報とに基づいて、前記アプリケーションに対して起動判断に関するメッセージを送信する手段を備えたことを特徴とする装置。
【請求項２】
前記アプリケーションが使用するリソースに関する情報を取得して前記リソース使用情報を生成するリソース使用情報取得手段を更に備えたことを特徴とする請求項１に記載の装置。
【請求項３】
前記リソース使用情報取得手段は、前記アプリケーションの初回の起動の際に、前記アプリケーションが使用するリソースに関する情報を取得して前記リソース使用情報を生成することを特徴とする請求項２に記載の装置。
【請求項４】
前記リソース使用情報取得手段は、前記アプリケーションの実行中に、定期的に前記アプリケーションが使用するリソースの使用量を取得し、定期的に取得した使用量に基づき前記リソース使用情報を生成し、保持することを特徴とする請求項２に記載の装置。
【請求項５】
前記リソース使用情報取得手段は、前記装置におけるプロセスが使用するリソースに関する情報を保持するシステム情報から前記アプリケーションが使用するリソースに関する情報を取得し、前記リソース使用情報を生成することを特徴とする請求項２ないし４のうちのいずれか１項に記載の装置。
【請求項６】
前記アプリケーションが使用するリソースに関する情報を前記アプリケーションから取得して前記リソース使用情報を生成するリソース使用情報取得手段を更に備えたことを特徴とする請求項１に記載の装置。
【請求項７】
前記リソースに関する情報は前記装置の機器構成情報を含むことを特徴とする請求項６に記載の装置。
【請求項８】
前記リソース使用情報取得手段は、
前記アプリケーションが固定的に使用するリソースについては、前記リソースに関する情報を前記アプリケーションから取得し、
前記アプリケーションの実行の度に使用量が変動するリソースについては、前記アプリケーションの実行中に、前記アプリケーションが使用するリソースの使用量を取得することを特徴とする請求項２に記載の装置。
【請求項９】
前記装置は、前記リソース状態情報と前記リソース使用情報とに基づき、前記アプリケーションを起動するためのリソースが不足していると判断した場合に、前記アプリケーションに対して起動不可であることを示すメッセージを送信することを特徴とする請求項１に記載の装置。
【請求項１０】
前記メッセージは、前記アプリケーションの起動に必要なリソースがあるか否かの情報を含み、
前記アプリケーションは該メッセージに基づき起動を継続するか否かを判断することを特徴とする請求項１に記載の装置。
【請求項１１】
リソース状態情報として予め割り当てられたリソースの量を用いることを特徴とする請求

項１ないし１０のうちいずれか１項に記載の装置。
【請求項１２】
前記装置は、予め組み込まれたソフトウェアによりサービスを提供する組み込み装置であり、
前記起動されるアプリケーションは、該予め組み込まれたソフトウェアとは別に前記装置に搭載されるアプリケーションである請求項１に記載の装置。
【請求項１３】
前記装置は画像形成装置であり、該画像形成装置は、
画像形成処理におけるハードウェアリソースと、
該ハードウェアリソースの制御に関するサービスを複数のアプリケーションに共通に提供するコントロールサービス部と、
該コントロールサービス部をサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービス部とを備え、
該仮想アプリケーションサービス部は、前記メッセージを送信する手段を有する請求項１２に記載の装置。
【請求項１４】
リソースに関する情報を含むメッセージを生成する手段を有する装置に、該メッセージに応じた処理を実行させるプログラムであって、
前記メッセージを受信する手順と、
該メッセージが、該プログラムが実行不可であることを示すメッセージである場合に、該プログラムにより前記装置に確保されたリソースを解放し、該プログラムを終了する手順と
を前記装置に実行させるプログラム。
【請求項１５】
リソースに関する情報を含むメッセージを生成する手段を有する装置に、該メッセージに応じた処理を実行させるプログラムであって、
前記メッセージを受信する手順と、
前記メッセージに含まれるリソースに関する情報に基づき、前記プログラムの起動を継続するか起動を停止するかの判断を行う手順と
を前記装置に実行させるプログラム。
【請求項１６】
前記リソースに関する情報に所定のリソースが不足していることを示す情報が含まれている場合に、該所定のリソースを使用しない制限モードで前記プログラムに規定される処理を前記装置に実行させる請求項１５に記載のプログラム。
【請求項１７】
前記制限モードでの前記プログラムの実行を許容するか否かを前記装置の操作部を介してユーザに問い合わせる手順を更に有し、該制限モードでの実行が許容されたときに前記プログラムに規定される処理を前記装置に実行させる請求項１６に記載のプログラム。
【請求項１８】
前記リソースに関する情報に所定のリソースが不足していることを示す情報が含まれている場合に、該プログラムにより前記装置に確保されたリソースを解放し、該プログラムを終了する手順を前記装置に実行させる請求項１５に記載のプログラム。
【請求項１９】
請求項１４ないし１８のうちいずれか１項に記載のプログラムを記録したコンピュータ読み取り可能な記録媒体。
【請求項２０】
アプリケーションを実行させるためのリソースを備えた装置における処理方法であって、
前記アプリケーションが起動される際に、前記装置におけるリソースの状態を示すリソース状態情報を取得し、取得したリソース状態情報と、前記アプリケーションが使用するリ

ソースに関するリソース使用情報とに基づいて、前記アプリケーションに対して起動判断に関するメッセージを送信するステップを有することを特徴とする装置。
【請求項２１】
前記アプリケーションが使用するリソースに関する情報を取得して前記リソース使用情報を生成するリソース使用情報取得ステップを更に備えたことを特徴とする請求項２０に記載の方法。
【請求項２２】
前記リソース使用情報取得ステップにおいて、前記装置は、前記アプリケーションの初回の起動の際に、前記アプリケーションが使用するリソースに関する情報を取得して前記リソース使用情報を生成することを特徴とする請求項２１に記載の方法。
【請求項２３】
前記リソース使用情報取得ステップにおいて、前記装置は、前記アプリケーションの実行中に、定期的に前記アプリケーションが使用するリソースの使用量を取得し、定期的に取得した使用量に基づき前記リソース使用情報を生成し、保持することを特徴とする請求項２１に記載の方法。
【請求項２４】
前記リソース使用情報取得ステップにおいて、前記装置は、前記装置におけるプロセスが使用するリソースに関する情報を保持するシステム情報から前記アプリケーションが使用するリソースに関する情報を取得し、前記リソース使用情報を生成する請求項２１ないし２３のうちのいずれか１項に記載の方法。
【請求項２５】
前記アプリケーションが使用するリソースに関する情報を前記アプリケーションから取得して前記リソース使用情報を生成するリソース使用情報取得ステップを更に備えたことを特徴とする請求項２０に記載の方法。
【請求項２６】
前記リソースに関する情報は前記装置の機器構成情報を含む請求項２５に記載の方法。
【請求項２７】
前記リソース使用情報取得ステップにおいて、前記装置は、
前記アプリケーションが固定的に使用するリソースについては、前記リソースに関する情報を前記アプリケーションから取得し、
前記アプリケーションの実行の度に使用量が変動するリソースについては、前記アプリケーションの実行中に、前記アプリケーションが使用するリソースの使用量を取得する請求項２１に記載の方法。
【請求項２８】
前記装置は、前記リソース状態情報と前記リソース使用情報とに基づき、前記アプリケーションを起動するためのリソースが不足していると判断した場合に、前記アプリケーションに対して起動不可であることを示すメッセージを送信する請求項２０に記載の方法。
【請求項２９】
前記メッセージは、前記アプリケーションの起動に必要なリソースがあるか否かの情報を含み、
前記アプリケーションは該メッセージに基づき起動を継続するか否かを判断する請求項２０に記載の方法。
【請求項３０】
リソース状態情報として予め割り当てられたリソースの量を用いる請求項２０ないし２９のうちいずれか１項に記載の方法。
【請求項３１】
前記装置は、予め組み込まれたソフトウェアによりサービスを提供する組み込み装置であり、
前記起動されるアプリケーションは、該予め組み込まれたソフトウェアとは別に前記装置に搭載されるアプリケーションである請求項２０に記載の方法。

【請求項３２】
前記装置は画像形成装置であり、該画像形成装置は、
画像形成処理におけるハードウェアリソースと、
該ハードウェアリソースの制御に関するサービスを複数のアプリケーションに共通に提供するコントロールサービス部と、
該コントロールサービス部をサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービス部とを備え、
該仮想アプリケーションサービス部が、前記メッセージを送信する請求項３１に記載の方法。

【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
この発明は、アプリケーションが使用するリソース量に応じて当該アプリケーションの起動に関する判断を行う技術に関する。
【０００２】
【従来の技術】
近年では、プリンタ、コピー、ファクシミリ、スキャナなどの各装置の機能を１つの筐体内に収納した画像形成装置（以下、「複合機」という。）が知られている。この複合機は、１つの筐体内に表示部、印刷部および撮像部などを設けるとともに、プリンタ、コピーおよびファクシミリ装置にそれぞれ対応した３種類のソフトウェアを設け、これらのソフトウェアを切り替えることによって、当該装置をプリンタ、コピー、スキャナまたはファクシミリ装置として動作させるものである。
【０００３】
このような従来の複合機では、メモリなどの限られたリソース（資源）の範囲内でプリンタ、コピー、ファクシミリ、スキャナなどの各機能単位ですべてのアプリケーションプログラムが起動される。言い換えれば、従来の複合機では、すべてのアプリケーションプログラムを起動できる程度のリソースを用意しており、アプリケーションプログラムが起動不可能になる状況が生じる場合は想定されていない。
【０００４】
【特許文献１】
特開平１１－１１２７０１号公報
【０００５】
【発明が解決しようとする課題】
ところで、このような従来の複合機では、プリンタ、コピー、スキャナおよびファクシミリ装置に対応するソフトウェアをそれぞれ別個に設けているため、各ソフトウェアの開発に多大の時間を要する。このため、出願人は、表示部、印刷部および撮像部などの画像形成処理で使用されるハードウェア資源を有し、プリンタ、コピーまたはファクシミリなどの各ユーザサービスにそれぞれ固有の処理を行うアプリケーションを複数搭載し、これらのアプリケーションとハードウェア資源との間に介在して、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の管理、実行制御並びに画像形成処理を行う各種コントロールサービスからなるプラットホームを備えた画像形成装置（複合機）を発明した。
【０００６】
このような新規な複合機では、アプリケーションの少なくとも２つが共通的に必要とするサービスを提供するコントロールサービスをアプリケーションと別個に設けた構成となっているため、アプリケーションのサイズが従来の複合機で動作するアプリケーションプログラムのサイズに比べて小さなものとなっており、アプリケーションの起動および終了が頻繁に行われる。
【０００７】

このため、複合機に搭載されているメモリなどのリソースの使用状況は頻繁に変化し、すべてのアプリケーションを起動した状態とすることができない場合も生じてくる。このような状況でアプリケーションを起動しても、アプリケーションが不正に終了してしまい、複合機の動作が不安定になるという問題がある。

【０００８】

また、かかる新規な複合機は、アプリケーションとコントロールサービスとを別個に設けているため、複合機の出荷後にユーザもしくは第三者であるサードベンダが新規なアプリケーションを開発して複合機に搭載可能な構成となっている。このため、複合機には、その出荷時に搭載されるコピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるアプリケーションの他、従来の複合機と異なり、このようなユーザやサードベンダが開発した新規アプリケーションなど、アプリケーションが多数起動可能となっている。コピー、プリンタ、ファクシミリ、スキャナなどの複合機であらかじめ提供されるアプリケーションの場合、限りのあるリソースを意識した構造となっているが、第三者が開発する新規アプリケーションの場合には、開発するベンダによって必ずしもリソースを意識した構造となっていない場合も考えられる。このため、限りのあるリソースの範囲内で新規アプリケーションの起動を行うと複合機が不安定になる可能性が高いという従来の複合機では問題にならなかった新規な課題が生じてくる。

【０００９】

この発明は上記に鑑みてなされたもので、アプリケーションの起動の際に、アプリケーションが使用するリソースと装置のリソースの状態とによって起動の可否を判断することにより、アプリケーションを実行する装置の安定性を図る技術を提供することを目的とする。

【００１０】

【課題を解決するための手段】

上記目的を達成するため、請求項１にかかる発明は、アプリケーションを実行させるためのリソースを備えた装置であって、前記アプリケーションが起動される際に、前記装置におけるリソースの状態を示すリソース状態情報を取得し、取得したリソース状態情報と、前記アプリケーションが使用するリソースに関するリソース使用情報とに基づいて、前記アプリケーションに対して起動判断に関するメッセージを送信する手段を備える。

【００１１】

本発明によれば、リソース状態情報とリソース使用情報とに基づき、アプリケーションに対して起動判断に関するメッセージを送信することができるので、当該メッセージに応じてアプリケーションは起動を継続するか否かを決定できる。従って、リソースが不足しているにもかかわらずにアプリケーションを起動してしまうことにより生じる不都合を防止でき、装置の安定性を向上させることができる。

【００１２】

請求項２に記載の発明は、請求項１の記載において、前記アプリケーションが使用するリソースに関する情報を取得して前記リソース使用情報を生成するリソース使用情報取得手段を更に備えるものである。本発明によれば、リソース使用情報取得手段を用いて例えばリソースの使用量の実績値を取得することができる。

【００１３】

請求項３に記載の発明は、請求項２の記載において、前記リソース使用情報取得手段は、前記アプリケーションの初回の起動の際に、前記アプリケーションが使用するリソースに関する情報を取得して前記リソース使用情報を生成するものである。

【００１４】

本発明によれば、２回目以降のアプリケーションの起動に際してリソース使用情報を使用できるようになる。

【００１５】

請求項４に記載の発明は、請求項２の記載において、前記リソース使用情報取得手段は、前記アプリケーションの実行中に、定期的に前記アプリケーションが使用するリソースの

使用量を取得し、定期的に取得した使用量に基づき前記リソース使用情報を生成し、保持するものである。

【００１６】

本発明によれば、定期的に取得した使用量に基づきリソース使用情報を生成するので、より的確なリソース使用情報を得ることができる。

【００１７】

請求項５に記載の発明は、請求項２ないし４のうちのいずれか１項の記載において、前記リソース使用情報取得手段は、前記装置におけるプロセスが使用するリソースに関する情報を保持するシステム情報から前記アプリケーションが使用するリソースに関する情報を取得し、前記リソース使用情報を生成する。

【００１８】

請求項６に記載の発明は、請求項１の記載において、前記アプリケーションが使用するリソースに関する情報を前記アプリケーションから取得して前記リソース使用情報を生成するリソース使用情報取得手段を更に備えるものである。

【００１９】

本発明によれば、アプリケーションから申告されたリソースに関する情報を使用できるので、種々の情報を起動の判断のために使用することが可能となる。

【００２０】

請求項７に記載の発明は、請求項６の記載において、前記リソースに関する情報は前記装置の機器構成情報を含むものである。本発明により、機器構成情報を起動の判断に用いることが可能となる。

【００２１】

請求項８に記載の発明は、請求項２の記載において、前記リソース使用情報取得手段は、前記アプリケーションが固定的に使用するリソースについては、前記リソースに関する情報を前記アプリケーションから取得し、前記アプリケーションの実行の度に使用量が変動するリソースについては、前記アプリケーションの実行中に、前記アプリケーションが使用するリソースの使用量を取得する。本発明によれば、効率的にリソース使用情報を取得できる。

【００２２】

請求項９に記載の発明は、請求項２の記載において、前記装置は、前記リソース状態情報と前記リソース使用情報とに基づき、前記アプリケーションを起動するためのリソースが不足していると判断した場合に、前記アプリケーションに対して起動不可であることを示すメッセージを送信するものである。

【００２３】

請求項１０に記載の発明は、請求項１の記載において、前記メッセージは、前記アプリケーションの起動に必要なリソースがあるか否かの情報を含み、前記アプリケーションは該メッセージに基づき起動を継続するか否かを判断するものである。

【００２４】

上記発明によれば、リソースが不足しているにもかかわらず、アプリケーションの起動を継続することにより生じる問題を回避できる。

【００２５】

請求項１１に記載の発明は、請求項１ないし１０のうちいずれか１項の記載において、リソース状態情報として予め割り当てられたリソースの量を用いることとしたものである。

【００２６】

請求項１２に記載の発明は、請求項１の記載において、前記装置は、予め組み込まれたソフトウェアによりサービスを提供する組み込み装置であり、前記起動されるアプリケーションは、該予め組み込まれたソフトウェアとは別に前記装置に搭載されるアプリケーションであるとしたものである。

【００２７】

本発明によれば、組み込み装置にアプリケーションを追加する際に、リソース量に応じた

起動判断を行うことが可能となる。

【００２８】

請求項１３に記載の発明は、請求項１２の記載において、前記装置は画像形成装置であり、該画像形成装置は、画像形成処理におけるハードウェアリソースと、該ハードウェアリソースの制御に関するサービスを複数のアプリケーションに共通に提供するコントロールサービス部と、該コントロールサービス部をサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービス部とを備え、該仮想アプリケーションサービス部は、前記メッセージを送信する手段を有するものである。

【００２９】

本発明によれば、画像形成装置にアプリケーションを追加する際に、リソース量に応じた起動判断を行うことが可能となる。

【００３０】

請求項１４に記載の発明は、リソースに関する情報を含むメッセージを生成する手段を有する装置に、該メッセージに応じた処理を実行させるプログラムであって、前記メッセージを受信する手順と、該メッセージが、該プログラムが実行不可であることを示すメッセージである場合に、該プログラムにより前記装置に確保されたリソースを解放し、該プログラムを終了する手順とを前記装置に実行させるプログラムである。

【００３１】

本発明によれば、リソースに関する情報を含むメッセージを送る手段を備えた装置に搭載するのに適したアプリケーションプログラムを提供できる。

【００３２】

請求項１５に記載の発明は、リソースに関する情報を含むメッセージを生成する手段を有する装置に、該メッセージに応じた処理を実行させるプログラムであって、前記メッセージを受信する手順と、前記メッセージに含まれるリソースに関する情報に基づき、前記プログラムの起動を継続するか起動を停止するかの判断を行う手順とを前記装置に実行させるプログラムである。

【００３３】

本発明によっても、リソースに関する情報を含むメッセージを送る手段を備えた装置に搭載するのに適したアプリケーションプログラムを提供できる。

【００３４】

請求項１６に記載の発明は、請求項１５の記載において、前記リソースに関する情報に所定のリソースが不足していることを示す情報が含まれている場合に、該所定のリソースを使用しない制限モードで前記プログラムに規定される処理を前記装置に実行させるプログラムである。本発明によれば、所定のリソースをしない制限モードで動作するアプリケーションを提供できる。

【００３５】

請求項１７に記載の発明は、請求項１６の記載において、前記制限モードでの前記プログラムの実行を許容するか否かを前記装置の操作部を介してユーザに問い合わせる手順を更に有し、該制限モードでの実行が許容されたときに前記プログラムに規定される処理を前記装置に実行させるプログラムである。本発明によりユーザの利便性が向上する。

【００３６】

請求項１８に記載の発明は、請求項１５の記載において、前記リソースに関する情報に所定のリソースが不足していることを示す情報が含まれている場合に、該プログラムにより前記装置に確保されたリソースを解放し、該プログラムを終了する手順を前記装置に実行させるプログラムである。請求項１９に記載の発明は、上記のプログラムを記録したコンピュータ読み取り可能な記録媒体である。請求項２０～３２に記載の発明は、上記の装置の発明に対応する方法の発明である。

【００３７】

【発明の実施の形態】

以下に添付図面を参照して、本発明の実施の形態である画像形成装置について詳細に説明する。

（実施の形態１）

図１は、この発明の実施の形態１である画像形成装置（以下、「複合機」という）の構成を示すブロック図である。図１に示すように、複合機１００は、白黒ラインプリンタ（Ｂ＆Ｗ　ＬＰ）１０１と、カラーラインプリンタ（Ｃｏｌｏｒ　ＬＰ）１０２と、スキャナ、ファクシミリ、ハードディスク、メモリ（ＲＡＭ、ＲＯＭなど）、ネットワークインタフェースなどのハードウェアリソース１０３を有するとともに、プラットホーム１２０とアプリケーション１３０と仮想アプリケーションサービス（ＶＡＳ：Ｖｅｒｔｉａｌ　Ａｐｐｌｉｃａｔｉｏｎ　Ｓｅｒｖｉｃｅ）１４０から構成されるソフトウェア群１１０とを備えている。

【００３８】

仮想アプリケーションサービス（ＶＡＳ）１４０は、アプリケーション１３０とプラットホーム１２０の間に配置される。ＶＡＳ１４０は、アプリケーション１３０の各アプリが初めて起動されたときに、各アプリが使用するリソースの使用状況を取得して、リソース使用情報ファイルをハードディスク（ＨＤ）に生成する。取得するリソースとしては、メモリに確保されるテキストメモリ領域サイズ、ヒープ領域サイズ、スタック領域サイズである。ここで、テキストメモリ領域とは、アプリケーション１３０の各アプリのプログラムがロードされるメモリ領域である。ヒープ領域とは、各アプリが動的に確保するメモリ領域であり、各アプリが使用する作業領域を含む。スタック領域とは、各アプリが実行されるとき、または各アプリが内部のモジュールを呼び出すときに使用する引数などを格納するために確保される領域である。

【００３９】

また、ＶＡＳ１４０は、各アプリの起動が２回目以降の場合、リソース使用情報ファイルからアプリが必要とする各リソースの容量を取得するとともに、各リソースの残容量を取得して、両者を比較し、アプリの起動の可否を判断する。具体的には、リソースの残容量がアプリが必要とする各リソースの容量より小さい場合には、起動されたアプリに対して起動停止要求メッセージを送信する。

【００４０】

なお、図１は、全てのアプリケーションに対してＶＡＳ１４０が配置される例を示しているが、図２に示すように、新規アプリケーションにのみに対してＶＡＳ１４０を配置するようにしてもよい。

【００４１】

プラットホーム１２０は、アプリケーションからの処理要求を解釈してハードウェア資源の獲得要求を発生させるコントロールサービスと、一または複数のハードウェア資源の管理を行い、コントロールサービスからの獲得要求を調停するシステムリソースマネージャ（ＳＲＭ）１２３と、汎用ＯＳ１２１とを有する。

【００４２】

コントロールサービスは、複数のサービスモジュールから形成され、ＳＣＳ（システムコントロールサービス）１２２と、ＥＣＳ（エンジンコントロールサービス）１２４と、ＭＣＳ（メモリコントロールサービス）１２５と、ＯＣＳ（オペレーションパネルコントロールサービス）１２６と、ＦＣＳ（ファックスコントロールサービス）１２７と、ＮＣＳ（ネットワークコントロールサービス）１２８とから構成される。なお、このプラットホーム１２０は、あらかじめ定義された関数により前記アプリケーション１３０から処理要求を受信可能とするアプリケーションプログラムインタフェース（ＡＰＩ）を有する。

【００４３】

汎用ＯＳ１２１は、ＵＮＩＸ（登録商標）などの汎用オペレーティングシステムであり、プラットホーム１２０並びにアプリケーション１３０の各ソフトウェアをそれぞれプロセスとして並列実行する。

【００４４】

ＳＲＭ１２３のプロセスは、ＳＣＳ１２２とともにシステムの制御およびリソースの管理を行うものである。ＳＲＭ１２３のプロセスは、スキャナ部やプリンタ部などのエンジン、メモリ、ＨＤＤファイル、ホストＩ／Ｏ（セントロＩ／Ｆ、ネットワークＩ／Ｆ、ＩＥＥＥ１３９４　Ｉ／Ｆ、ＲＳ２３２Ｃ　Ｉ／Ｆなど）のハードウェア資源を利用する上位層からの要求にしたがって調停を行い、実行制御する。

【００４５】

具体的には、このＳＲＭ１２３は、要求されたハードウェア資源が利用可能であるか（他の要求により利用されていないかどうか）を判断し、利用可能であれば要求されたハードウェア資源が利用可能である旨を上位層に伝える。また、ＳＲＭ１２３は、上位層からの要求に対してハードウェア資源の利用スケジューリングを行い、要求内容（例えば、プリンタエンジンにより紙搬送と作像動作、メモリ確保、ファイル生成など）を直接実施している。

【００４６】

ＳＣＳ１２２のプロセスは、アプリ管理、操作部制御、システム画面表示、ＬＥＤ表示、リソース管理、割り込みアプリ制御などを行う。

【００４７】

ＥＣＳ１２４のプロセスは、白黒ラインプリンタ（Ｂ＆Ｗ　ＬＰ）１０１、カラーラインプリンタ（Ｃｏｌｏｒ　ＬＰ）１０２、スキャナ、ファクシミリなどからなるハードウェアリソース１０３のエンジンの制御を行う。

【００４８】

ＭＣＳ１２５のプロセスは、画像メモリの取得および解放、ハードディスク装置（ＨＤＤ）の利用、画像データの圧縮および伸張などを行う。

【００４９】

ＦＣＳ１２７のプロセスは、システムコントローラの各アプリ層からＰＳＴＮ／ＩＳＤＮ網を利用したファクシミリ送受信、ＢＫＭ（バックアップＳＲＡＭ）で管理されている各種ファクシミリデータの登録／引用、ファクシミリ読みとり、ファクシミリ受信印刷、融合送受信を行うためのＡＰＩを提供する。

【００５０】

ＮＣＳ１２８のプロセスは、ネットワークＩ／Ｏを必要とするアプリケーションに対して共通に利用できるサービスを提供するためのプロセスであり、ネットワーク側から各プロトコルによって受信したデータを各アプリケーションに振り分けたり、アプリケーションからデータをネットワーク側に送信する際の仲介を行う。具体的には、ｆｔｐｄ、ｈｔｔｐｄ、ｌｐｄ、ｓｎｍｐｄ、ｔｅｌｎｅｔｄ、ｓｍｔｐｄなどのサーバデーモンや、同プロトコルのクライアント機能などを有している。

【００５１】

ＯＣＳ１２６のプロセスは、オペレータ（ユーザ）と本体制御間の情報伝達手段となるオペレーションパネル（操作パネル）の制御を行う。ＯＣＳ１２６は、オペレーションパネルからキー押下をキーイベントとして取得し、取得したキーに対応したキーイベント関数をＳＣＳ１２２に送信するＯＣＳプロセスの部分と、アプリケーション１３０またはコントロールサービスからの要求によりオペレーションパネルに各種画面を描画出力する描画関数やその他オペレーションパネルに対する制御を行う関数などがあらかじめ登録されたＯＣＳライブラリの部分とから構成される。このＯＣＳライブラリは、アプリケーション１３０およびコントロールサービスの各モジュールにリンクされて実装されている。なお、ＯＣＳ１２６のすべてをプロセスとして動作させるように構成しても良く、あるいはＯＣＳ１２６のすべてをＯＣＳライブラリとして構成しても良い。

【００５２】

アプリケーション１３０は、ページ記述言語（ＰＤＬ）、ＰＣＬおよびポストスクリプト（ＰＳ）を有するプリンタ用のアプリケーションであるプリンタアプリ１１１と、コピー用アプリケーションであるコピーアプリ１１２と、ファクシミリ用アプリケーションであるファックスアプリ１１３と、スキャナ用アプリケーションであるスキャナアプリ１１４

と、ネットワークファイル用アプリケーションであるネットファイルアプリ１１５と、工程検査用アプリケーションである工程検査アプリ１１６とを有している。これらの各アプリは、その起動時にＶＡＳ１４０に対して自プロセスのプロセスＩＤとともにアプリ登録要求メッセージを送信し、アプリ登録要求メッセージを受信したＶＡＳ１４０によって、起動したアプリに対する登録処理が行われるようになっている。なお、図２に示す構成の場合には、ＶＡＳ１４０が配置されていないアプリケーションは、アプリ登録要求メッセージをＳＣＳ１２２に対して送信する。

【００５３】

アプリケーション１３０の各プロセス、コントロールサービスの各プロセスは、関数呼び出しとその戻り値送信およびメッセージの送受信によってプロセス間通信を行いながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを実現している。

【００５４】

このように、実施の形態１にかかる複合機１００には、複数のアプリケーション１３０および複数のコントロールサービスが存在し、いずれもプロセスとして動作している。そして、これらの各プロセス内部には、一または複数のスレッドが生成されて、スレッド単位の並列実行が行われる。そして、コントロールサービスがアプリケーション１３０に対し共通サービスを提供しており、このため、これらの多数のプロセスが並列動作、およびスレッドの並列動作を行って互いにプロセス間通信を行って協調動作をしながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを提供するようになっている。また、複合機１００には、サードベンダなどの第三者がコントロールサービス層の上のアプリケーション層に新規アプリ１１７，１１８を開発して搭載することが可能となっている。図１、図２では、この新規アプリ１１７，１１８を搭載した例を示している。

【００５５】

なお、実施の形態１にかかる複合機１００では、複数のアプリケーション１３０のプロセスと複数のコントロールサービスのプロセスとが動作しているが、アプリケーション１３０とコントロールサービスのプロセスがそれぞれ単一の構成とすることも可能である。また、各アプリケーション１３０は、アプリケーションごとに追加または削除することができる。

【００５６】

図３は、実施の形態１にかかる複合機１００のＶＡＳ１４０の構成と、ＶＡＳ１４０と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図３では、アプリケーション１３０の例として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ１１７，１１８を示しているが、他のアプリでも同様の構成である。

【００５７】

仮想アプリケーションサービス（ＶＡＳ）１４０のプロセスには、ディスパッチャ１４４と、制御スレッド１４３と、リソース使用情報取得スレッド１４１と、アプリ起動判断スレッド１４２とが動作している。

【００５８】

ディスパッチャ１４４は、アプリケーション１３０やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じて制御スレッド１４３、リソース使用情報取得スレッド１４１、アプリ起動判断スレッド１４２に処理要求を行うものである。実施の形態１の複合機１００では、ディスパッチャ１４４は、各アプリが起動する際にアプリからアプリ登録要求メッセージを受信したとき、受信したアプリ登録要求メッセージを制御スレッド１４３に送信するようになっている。

【００５９】

制御スレッド１４３は、ディスパッチャ１４４からアプリ登録要求メッセージを受信してアプリ登録処理を行う。ここで、アプリ登録処理とは、ＲＡＭ２１０にアプリ登録テーブ

ル（図示せず）を生成し、アプリ登録要求メッセージを送信したアプリの識別情報であるアプリＩＤをアプリ登録テーブルに記録する処理をいう。

【００６０】

また、制御スレッド１４３は、ＨＤ２００に格納されたリソース使用情報ファイル２０１を参照して、アプリ登録要求を行ったアプリについて、リソース使用情報が記録されているか否かをチェックすることにより、アプリの起動が初回の起動か２回目以降の起動かを判断する。そして、初回の起動である場合にはリソース使用情報取得スレッド１４１に対してアプリＩＤ、アプリのプロセスＩＤとともにリソース使用情報取得処理要求を行い、２回目以降の起動である場合には、アプリ起動判断スレッド１４２に対してアプリＩＤ、アプリのプロセスＩＤとともにアプリ起動判断処理要求を行う。

【００６１】

リソース使用情報取得スレッド１４１は、制御スレッド１４３からの処理要求を受けると、汎用ＯＳ１２１が管理するＲＡＭ２１０上のｐｒｏｃ構造体２１１（またはｕ領域２１２）を参照して、各アプリが使用するテキストメモリ領域サイズ、ヒープ領域サイズ、スタック領域サイズ、ＣＰＵ占有時間を取得して、リソース使用情報ファイル２０１としてハードディスクに生成する。かかるリソース使用情報はアプリごとのレコードとして記録される。なお、上記のＣＰＵ占有時間とは、例えば最近のＣＰＵ占有時間である。

【００６２】

上記ＶＡＳ１４０のプログラムは、ソフトウェア開発キット（ＳＤＫ：Ｓｏｆｔｗａｒｅ　Ｄｅｖｅｌｏｐｍｅｎｔ　Ｋｉｔ）等の一部または全部として、ＣＤ－ＲＯＭまたはＦＤ（フレキシブルディスク）などの記憶媒体に実行可能な形式またはインストール可能な形式のファイルで提供される。また、このような実行可能な形式またはインストール可能な形式のＶＡＳ１４０のプログラムファイルを、ネットワーク経由で取得可能な方法で提供するようにしても良い。なお、複合機１００にインストールされるアプリケーションについてもＣＤ－ＲＯＭまたはＦＤ（フレキシブルディスク）などの記憶媒体に格納して提供できる。更に、ネットワーク経由で取得可能な方法で提供するようにしても良い。

【００６３】

図４は、ＨＤ２００に格納されるリソース使用情報ファイル２０１の内容例を示す説明図である。図４に示すように、リソース使用情報ファイル２０１には、アプリＩＤごとに、テキストメモリサイズ、ヒープサイズ、スタックサイズ、ＣＰＵ占有時間が記録されている。

【００６４】

図５は、リソース使用情報取得スレッド１４１が参照するｐｒｏｃ構造体２１１の一例を示す説明図である。図５に示すように、ｐｒｏｃ構造体２１１には、各プロセスごとに、プロセスＩＤ（ｐ＿ｐｉｄ）、ＣＰＵ占有時間、テキストメモリサイズ、ヒープメモリサイズ、スタックサイズなどが格納されている。このｐｒｏｃ構造体２１１は、プロセス実行時、プロセス終了時、およびプロセスの状態が変化したときに汎用ＯＳ１２１によって更新される。ｐｒｏｃ構造体２１１の各情報は、ＶＡＳ１４０からのシステムコールによってＶＡＳ１４０が取得することができる。

【００６５】

アプリ起動判断スレッド１４２は、アプリの起動が２回目以降である場合に、リソース使用情報ファイル２０１を参照して、アプリ登録要求を行ったアプリが必要とする各リソースの容量を取得する。また、アプリ起動判断スレッド１４２は、汎用ＯＳ１２１のシステムコールあるいはコントロールサービスで提供されるサービス関数呼び出しにより、ｐｒｏｃ構造体２１１を参照することにより、画像メモリ領域の残容量、ヒープ領域の残容量、スタック領域の残容量を取得する。そして、両者を比較して、アプリ起動の可否を判断する。また、アプリ起動判断スレッド１４２は、汎用ＯＳ１２１のシステムコールあるいはコントロールサービスで提供されるサービス関数呼び出しにより、ＣＰＵ稼働率を取得する。

【００６６】

そして、ＣＰＵ稼働率が一定値を超えているか否か、およびＣＰＵ占有時間が一定時間を超えているか否かによってアプリ起動の可否を判断する。なお、ＣＰＵ占有時間とＣＰＵ稼働率との関係、すなわち、ＣＰＵ稼働率の値に応じてアプリ起動すべきと判断するＣＰＵ占有時間の関係は予め定めておくが、複合機１００によって任意に定めることができる。

【００６７】

アプリ起動判断スレッド１４２は、アプリ起動が可能と判断した場合には、アプリ登録要求をしたアプリに対して起動可能メッセージを送信する。一方、アプリ起動を続行すべきでないと判断した場合には、アプリ登録要求をしたアプリに対して起動終了要求メッセージを送信する。

【００６８】

次に、このように構成された複合機１００のＶＡＳ１４０によるアプリ起動判断処理について説明する。図６は、ＶＡＳ１４０の制御スレッド１４３によるアプリ登録およびアプリ起動回数判断の処理手順を示すフローチャートである。

【００６９】

ディスパッチャ１４４が起動する際に、アプリからアプリ登録要求メッセージを受信すると、アプリ登録要求メッセージをそのアプリのプロセスＩＤとともに制御スレッド１４３に受け渡す。制御スレッド１４３は、アプリ登録要求メッセージとプロセスＩＤをディスパッチャ１４４から受信すると（ステップＳ５０１）、アプリを識別するアプリＩＤを決定し、アプリ登録テーブル（図示せず）にアプリＩＤを記録することにより、アプリ登録を行う（ステップＳ５０２）。なお、アプリＩＤは、コピーアプリ１１２、プリンタアプリ１１１など既存のアプリケーションについては、予め定められており、各アプリＩＤをＶＡＳ１４０が内部で保持している。また、サードベンダなどが開発した新規アプリ１１７，１１８については、最初の起動時におけるアプリ登録処理の中で決定される。

【００７０】

そして、制御スレッド１４３は、ＨＤ２００に格納されているリソース使用情報ファイル２０１を参照し（ステップＳ５０３）、リソース使用情報ファイル２０１の中にアプリ登録処理で登録したアプリＩＤのリソース使用情報が格納されているか否かをチェックすることにより、アプリ登録要求を行ったアプリが最初の起動か、２回目以降の起動かを判断する（ステップＳ５０４）。

【００７１】

そして、リソース使用情報ファイル２０１の中にアプリ登録要求を行ったアプリのアプリＩＤに対するリソース使用情報が格納されていない場合には（ステップＳ５０４：Ｎｏ）、初回の起動であると判断し、アプリが使用するリソースのリソース使用情報を取得するため、リソース使用情報取得スレッド１４１に対して、アプリＩＤおよびアプリのプロセスＩＤとともにリソース使用情報取得要求メッセージを送信する（ステップＳ５０５）。

【００７２】

一方、リソース使用情報ファイル２０１の中にアプリ登録要求を行ったアプリのアプリＩＤに対するリソース使用情報が格納されている場合には（ステップＳ５０４：Ｙｅｓ）、２回目以降の起動であると判断し、アプリ起動の可否を判断するため、アプリ起動判断スレッド１４２に対し、アプリＩＤおよびアプリのプロセスＩＤとともにアプリ起動判断要求メッセージを送信する（ステップＳ５０６）。

【００７３】

図７は、リソース使用情報取得スレッド１４１によるリソース使用情報取得の処理手順を示すフローチャートである。リソース使用情報取得スレッド１４１では、アプリの最初の起動時に次の処理が実行される。

【００７４】

リソース使用情報取得スレッド１４１は、アプリＩＤ、プロセスＩＤとリソース使用情報取得要求メッセージを制御スレッド１４３から受信すると（ステップＳ６０１）、ｐｒｏｃ構造体２１１を参照して該当するアプリＩＤのプロセスＩＤのブロックの位置を検索す

る（ステップＳ６０２）。そして、検索されたプロセスＩＤのブロックから、テキストメモリ領域サイズ、ヒープ領域サイズ、スタック領域サイズおよびＣＰＵ占有時間のリソース使用情報を取得する（ステップＳ６０３）。そして、取得したこれらのリソース使用情報を、アプリＩＤとともにリソース使用情報ファイル２０１に記録する（ステップＳ６０４）。これにより、アプリが起動時に必要となるリソースの情報がリソース使用情報ファイル２０１に格納されることになる。

【００７５】

図８は、アプリ起動判断スレッド１４２によるアプリ起動可否判断の処理手順を示すフローチャートである。アプリ起動判断スレッド１４２では、アプリの２回目以降の起動時に次の処理が実行される。

【００７６】

アプリ起動判断スレッド１４２は、アプリＩＤ、プロセスＩＤとアプリ起動判断要求メッセージを制御スレッド１４３から受信すると（ステップＳ７０１）、ＨＤ２００に格納されているリソース使用情報ファイル２０１から該当するアプリＩＤのリソース使用情報のレコードを検索し、テキストメモリ領域サイズ、ヒープ領域サイズ、スタック領域サイズおよびＣＰＵ占有時間を取得する（ステップＳ７０２）。次に、アプリ起動判断スレッド１４２は、システムコールあるいはコントロールサービス関数の呼び出しによって、ｐｒｏｃ構造体から各プロセスによるリソースの使用量を取得することにより、現在におけるメモリの残容量を得る（ステップＳ７０３）。このとき、取得される残容量は、テキストメモリ領域の残容量、ヒープ領域の残容量、スタック領域の残容量である。

【００７７】

そして、アプリ起動判断スレッド１４２は、リソース使用情報ファイル２０１から取得したアプリ起動に必要なテキストメモリ領域サイズとテキストメモリ領域の残容量とを比較する（ステップＳ７０４）。比較の結果、テキストメモリ領域サイズがテキストメモリ領域の残容量より大きい場合には（ステップＳ７０４：Ｎｏ）、アプリを実行することができないと判断して、アプリに対してアプリ起動終了要求メッセージを送信する（ステップＳ７０９）。

【００７８】

一方、テキストメモリ領域サイズがテキストメモリ領域の残容量以下の場合には（ステップＳ７０４：Ｙｅｓ）、アプリ実行に必要なテキストメモリ領域を現在の状況で確保できると判断する。そして、次に、リソース使用情報ファイル２０１から取得したアプリの実行に必要なヒープ領域サイズとヒープ領域の残容量とを比較する（ステップＳ７０５）。比較の結果、ヒープ領域サイズがヒープ領域の残容量より大きい場合には（ステップＳ７０５：Ｎｏ）、アプリを実行することができないと判断して、アプリに対してアプリ起動終了要求メッセージを送信する（ステップＳ７０９）。

【００７９】

一方、ヒープ領域サイズがヒープ領域の残容量以下の場合には（ステップＳ７０５：Ｙｅｓ）、アプリ実行に必要なヒープ領域を現在の状況で確保できると判断する。そして、次に、リソース使用情報ファイル２０１から取得したアプリの実行に必要なスタック領域サイズとスタック領域の残容量とを比較する（ステップＳ７０６）。比較の結果、スタック領域サイズがスタック領域の残容量より大きい場合には（ステップＳ７０６：Ｎｏ）、アプリを実行することができないと判断して、アプリに対してアプリ起動終了要求メッセージを送信する（ステップＳ７０９）。

【００８０】

一方、スタック領域サイズがスタック領域の残容量以下の場合には（ステップＳ７０６：Ｙｅｓ）、アプリ実行に必要なスタック領域を現在の状況で確保できると判断する。そして、次に、システムコールの発行によって、現在のＣＰＵ稼働率を取得する（ステップＳ７０７）。そして、予め定めたＣＰＵ稼働率に対して許容された時間内にＣＰＵ占有時間が含まれるか否かを判断する（ステップＳ７０８）。そして、許容された時間外である場合（ステップＳ７０８：Ｎｏ）、例えば、ＣＰＵ稼働率５０～６０％のとき、ＣＰＵ占有

時間が６０ｍｓ以下の場合にはアプリ起動を続行する対応関係を定めている場合において、ＣＰＵ稼働率５５％で、かつＣＰＵ占有時間が８０ｍｓである場合には、かかる状態でアプリを実行するとシステム全体が不安定になると判断して、アプリに対してアプリ起動終了要求メッセージを送信する（ステップＳ７０９）。

【００８１】

一方、ＣＰＵ占有時間が許容された時間内である場合（ステップＳ７０８：Ｙｅｓ）、たとえば上記例のＣＰＵ稼働率５５％でＣＰＵ占有時間が４０ｍｓの場合には、アプリ実行が安定動作すると判断する。これにより、すべてのリソースを確保できることがわかったので、次に、アプリ起動判断スレッド１４２は、アプリに対してアプリ起動可能通知メッセージを送信する（ステップＳ７１０）。

【００８２】

なお、各アプリは、アプリ起動判断スレッド１４２からアプリ起動可能通知メッセージを受信すると、そのまま処理実行を継続する。一方、各アプリは、アプリ起動判断スレッド１４２からアプリ起動終了要求メッセージを受信すると、ただちにアプリの実行を終了する。また、アプリ起動終了要求メッセージを受信したアプリが、それまでに確保していたリソースを解放してから実行を停止するようにしてもよい。

【００８３】

なお、アプリ起動判断スレッド１４２は、アプリの起動を停止すべきとの判断をした場合には、更に、ＭＣＳ１２５などのコントロールサービスに対して、テキストメモリ領域、ヒープ領域あるいはスタック領域を増大して確保する旨の要求メッセージを送信して、アプリ起動に必要なサイズを確保してからアプリに対して起動可能通知メッセージを送信するように構成しても良い。この場合には、単にアプリ起動終了要求を行う場合に比べて、ユーザの利便性の向上を図ることができる。

【００８４】

このように、実施の形態１にかかる複合機１００では、仮想アプリケーションサービス１４０のリソース使用情報取得スレッド１４１によって、アプリケーション１３０が使用するリソースに関する情報を取得してリソース使用情報ファイル２０１を生成し、アプリ起動判断スレッド１４２によって、アプリケーション１３０が起動されたときに、テキストメモリ領域残容量、ヒープ領域残容量、スタック領域残容量、ＣＰＵ稼働率を取得し、取得したこれらのリソースの状態と、リソース使用情報ファイル２０１の中に記録されたアプリケーション１３０が使用するリソースの情報とに基づいてアプリケーション１３０の起動可否を判断しているので、リソースの状況によってアプリケーション１３０の実行ができなくなることを未然に回避して、複合機１００のシステムの安定性を向上させることができる。また、サードベンダが開発した新規アプリ１１７，１１８がリソースの限界を意識した処理を行っていない場合でも、リソース不足に伴ってシステムが不安定になることを回避でき、複合機１００のシステムの安定性を向上させることができる。

【００８５】

なお、実施の形態１にかかる複合機１００では、ＶＡＳ１４０が全てのアプリケーション１３０に対してリソース使用情報取得処理、アプリ起動判断処理を行っているが、図２に示したように一部のアプリに対してのみかかる処理を行うように構成しても良い。例えば、新規アプリ１１７，１１８などサードベンダなどの第三者が開発したアプリにのみリソース使用情報取得処理、アプリ起動判断処理を行い、プリンタアプリ１１１やコピーアプリ１１２などの既存のアプリに対してはこのようなサービスを行わないように構成しても良い。

【００８６】

また、上記のように初回の起動時のリソースの使用量をリソース使用情報ファイル２０１に記録して後の起動判断に用いる他、アプリケーションが実行されているその実行期間（起動から終了まで）の中で定期的にｐｒｏｃ構造体を参照することによりリソースの使用量を複数回取得し、その中の平均値をリソース使用情報ファイル２０１に記録するようにしてもよい。また、複数回取得した中での最大値をリソース使用情報ファイル２０１に記

録するようにしてもよい。

【００８７】

また、上記の実施の形態のように、初回の実行時の値を求めて記録しておくことの他、アプリケーションの実行の度に上記の平均値を求め、その平均値が前回の平均値を超えた場合にリソース使用情報ファイル２０１を更新するようにしてもよい。また、アプリケーションの実行の度に上記の最大値を求め、その最大値が前回の最大値を超えた場合にリソース使用情報ファイル２０１を更新するようにしてもよい。更に、アプリケーションの実行毎の上記平均値を別に記録しておき、これらの中の平均値をリソース使用情報ファイル２０１に記録するようにしてもよい。また、アプリケーションの実行毎の上記最大値を別に記録しておき、これらの中の平均値又は最大値をリソース使用情報ファイル２０１に記録するようにしてもよい。

【００８８】

（実施の形態２）

次に本発明の実施の形態２について説明する。実施の形態１では、ＶＡＳ１４０がアプリケーションの起動可否を判断していたが、実施の形態２では、ＶＡＳ１４０からリソース量に関する判定メッセージをアプリケーションが受信することにより、アプリケーションが起動の可否等を判断することも可能な構成としている。また、メモリやＣＰＵのみでなく、複合機に接続されているユニットなどのシステム機器構成情報も起動の判断に使用することが可能となっている。

【００８９】

実施の形態２の構成は、図１、もしくは図２に示した実施の形態１の構成と同様である。また、以下説明するＶＡＳ１４０による処理は、図３に示したようにスレッドとして実行してもよいし、ＶＡＳ１４０のプロセスとして実行してもよい。

【００９０】

実施の形態１では、アプリの起動実績を記録したリソース使用情報ファイルをアプリケーションが使用を予定するリソース量を格納したファイルとして使用しているが、実施の形態２では、ＣＰＵ使用量のように使用量が動的に変動するリソースについては実績を記録したリソース使用情報ファイルを使用し、メモリ領域のようにアプリケーションが固定的に使用するリソースについては、アプリケーションから起動時に申告される使用リソース情報を使用する。なお、この使用リソース情報は、アプリケーションの初回の起動時にＶＡＳ１４０がアプリケーションから（より詳細には所定のメモリ領域から）取得し、アプリケーション管理ファイルとして複合機のハードディスクに記録されるものである。なお、固定的に使用するリソースとは、アプリケーションの実行の度における使用量の変化がないか、変化が少ないリソースのことである。

【００９１】

図９にアプリケーション管理ファイルの内容例を示す。同図に示すように、アプリケーション名、バージョンなどの情報と、メモリやシステム機器構成を含む使用リソース情報が記録されている。また、ＲＡＭ（本実施の形態では複合機のＮＶ－ＲＡＭ（ｎｏｎｖｏｌａｔｉｌｅ　ＲＡＭ）を使用する）にアプリケーション管理ファイルのＨＤ２００における場所を示す情報が格納される。

【００９２】

図１０に、新規アプリケーションに係るＮＶ－ＲＡＭの構成例とＨＤの構成例を示す。なお、ここでいう新規アプリケーションとは、図１、図２におけるアプリケーション１１７、１１８に対応するものである。

【００９３】

同図に示すように、ＮＶ－ＲＡＭには登録済みの新規アプリケーション毎に、アプリケーションのプロダクトＩＤ、使用ＮＶ－ＲＡＭサイズ、ＮＶ－ＲＡＭにおける使用領域の開始アドレスを示すオフセットなどが記録される。また、ＮＶ－ＲＡＭに記録されたプロダクトＩＤに対応してＨＤ２００に当該アプリケーション用の領域が設けられる。

【００９４】

例えば、アプリケーション１に対応して、ＨＤ２００にディレクトリが設けられ、その中に、アプリケーション１に対してシステム（ＶＡＳ１４０）が使用する領域とアプリケーション１自身が使用する領域が設けられる。図９に示したアプリケーション管理ファイルはシステム使用領域の中に格納される。

【００９５】

次に、アプリケーションの起動の際に、ＶＡＳ１４０が実行する処理を図１１のフローチャートを用いて説明する。

【００９６】

実施の形態１と同様に、アプリケーションが起動する際に、ＶＡＳ１４０がアプリ登録要求メッセージを受信すると（ステップＳ８０１）、当該アプリケーションに対応するアプリケーション管理ファイルがあるか否かをＨＤ２００にアクセスすることにより調べる（ステップＳ８０２）。アプリケーション管理ファイルがなければ、アプリケーションから申告される情報に基づきそれを作成する（ステップＳ８０３）。なお、アプリケーション管理ファイルがあるか否かはＮＶ－ＲＡＭにアクセスすることにより判断してもよい。

【００９７】

アプリケーション管理ファイルがあればそこから当該アプリケーションが使用する予定のリソース情報を取得する（ステップＳ８０４）。なお、アプリケーション管理ファイルを用いる代わりにアプリケーションから申告されるリソース情報を用いてもよい。また、ＣＰＵ使用量（実績値）に関しては、実施の形態１で説明したような方法でＣＰＵ使用量に関するリソース使用情報ファイルを作成しておき、そこから取得する。すなわち、ＣＰＵ使用量に関しては初回の起動時には判断に用いず、２回目以降の起動時に判断に用いる。なお、ＣＰＵ使用量を起動の判断に用いないようにしてもよい。また、リソース使用情報ファイルの内容を、アプリケーション管理ファイルに記録するようにしてもよい。

【００９８】

次に、現在の複合機１００におけるリソース情報を取得する（ステップＳ８０５）。そして、メモリ容量などのリソースに関しては現在の複合機１００における残容量を求めておく。そして、ステップＳ８０４にて取得した使用予定リソース量と現在のリソース量とを、使用予定リソースにおける個々のリソース毎に比較することにより、使用予定リソース量が現在のシステムリソース量を超えているか否かを調べる（ステップＳ８０６）。また、システム機器構成についてのリソース（ユニット等）については、アプリケーションが使用する予定のリソースが現在のリソースの中にあるか否かを調べる（ステップＳ８０６）。

【００９９】

使用予定リソース量が現在のシステムリソース量を超えておらず、かつ、不足しているシステム機器構成リソースがない場合には、アプリケーションの起動を継続する（ステップＳ８０７）。使用予定リソース量が現在のシステムリソース量を超えているリソースがある場合、もしくは、不足しているシステム機器構成リソースがある場合、リソース判定結果通知メッセージをアプリケーションに対して送信する（ステップＳ８０８）。このメッセージの内容に基づき、アプリケーションは起動を継続するか否かの判断を行う。なお、本実施の形態ではリソース判定結果通知メッセージは、使用予定リソース毎に、アプリケーションが必要とするシステム機器構成（ユニットなど）上のリソースについては当該リソースの有無（有りの場合ＯＫ、無しの場合ＮＧ）、また、メモリ領域などのリソースについては使用予定リソース量が現在のシステムリソース量を超えていなければＯＫ、超えていればＮＧを示す情報が含まれる。

【０１００】

次に、リソース判定結果通知メッセージを受信したアプリケーションの動作について図１２を参照して説明する。

【０１０１】

ＶＡＳ１４０からリソース判定結果通知メッセージを受信すると（ステップＳ９０１）、メッセージに含まれる最初のリソースについて、当該リソースがＮＧ判定か否かをチェッ

クする（ステップＳ９０２）。
【０１０２】
ＮＧ判定でなければ次のリソースがあるか否かをチェックして（ステップＳ９０３）、あれば次のリソースについてのチェックを行う。ステップＳ９０２でＮＧ判定であれば、当該リソースに関する制限モードがあれば当該リソースに関する使用不可フラグをアプリケーションに設定する（ステップＳ９０４のＹＥＳ、ステップＳ９０５）。なお、制限モードとしては、一部のシステム機器構成における機器などを使用しないこと、もしくは、所定のメモリ領域の使用量を削減することなどがあるが、図１２に示す例は、対象のリソースを使用しない場合について示している。
【０１０３】
この場合、アプリケーションが操作パネル上に当該リソースがＮＧ判定であることを表示し、制限モードでの動作を許容するか否かをユーザに問い合わせ、ユーザが制限モードでの動作を許容する場合のみ制限付きで起動を続行するようにしてもよい。
【０１０４】
また、ＣＰＵパワーが不足しているとの判定結果に対しては、その旨を操作パネルに表示し、動作が遅くなることを許容するか否かをユーザに問い合わせ、許容する場合に起動を継続するといった処理も可能である。
【０１０５】
ステップＳ９０３において次のリソース情報がない場合には、アプリケーションの起動を継続するか、もしくは制限付き起動手続きを行う（ステップＳ９０６）。
【０１０６】
ステップＳ９０４において制限動作モードがない場合には、アプリケーション起動終了手続きを行う（ステップＳ９０７）。図１３を用いてこの処理の手順について説明する。
【０１０７】
まず、操作部表示に関するリソースを開放する（ステップＳ１００１）。そして、これまでに確保したメモリを開放し（ステップＳ１００２）、ＶＡＳ１４０に対してアプリケーション登録抹消要求を行う（ステップＳ１００３）。次に、プロセス間通信の停止処理を行い（ステップＳ１００４）、システムコールによる自プロセスの終了を行う（ステップＳ１００５）。なお、必要に応じて上記の処理の前に、使用者への通知、スキャナ、プロッタの解放、ネットワークリソースの解放を行う。
【０１０８】
なお、図１１に示したステップＳ８０６において、使用予定リソース量が現在のシステムリソース量を超えているリソースがある場合であって、当該リソースがアプリケーションにとって必須の固定的に使用するメモリ領域（テキストメモリ領域、ヒープ領域、スタック領域）である場合には、ＶＡＳ１４０が当該アプリケーションは起動できないことを判断し、上記のリソース判定結果通知メッセージとして起動停止要求をアプリケーションに送信するようにしてもよい。この場合、アプリケーションはすぐに図１３に示した処理を実行する。また、ＶＡＳ１４０が、アプリケーションが制限付き起動をするかどうかを判断し、制限付き起動が可能と判断した場合には、その旨をアプリケーションに通知してもよい。この場合、ＶＡＳ１４０がユーザに制限付き起動を問い合わせるようにすることができる。
【０１０９】
上記のように起動の判断におけるリソース情報として、図９の「システム構成関連」として示したシステム機器構成に関する情報も用いることにより、例えば、パンチやステープル機能を有する機器が複合機１００に実装されていなければ使用できないアプリケーションを、そのような機器が複合機に取り付けられていないないにもかかわらず起動するといった無駄な起動を抑制できる。
【０１１０】
また、起動の判断に上記の情報に加えて機種情報を用いても良い。これにより、機種、モデルによる機能の差異に起因する機能不全や動作不良を抑制することが可能となる。

【０１１１】
なお、図１１のフローにおいて、ＶＡＳ１４０が、アプリケーションが使用を予定しているリソース情報の取得（ステップＳ８０４）を行わずに、アプリケーションに現状の機器構成情報と残存リソース量をそのままリソース判定結果通知メッセージとして送信し、アプリケーション側で必要リソースの判断を行うようにしてもよい。この場合、アプリケーションが保持する使用予定リソースを一つ一つリソース判定結果メッセージにおけるリソースと比較し、使用予定リソースの有無を判定し、起動の判断をアプリケーションが行う。
【０１１２】
また、図１１に示す処理は、アプリケーションの起動時毎に行ってもよいし、当該アプリケーションのインストール時のみに行う事もできる。
【０１１３】
次に、図１１のステップＳ８０５におけるＶＡＳ１４０による現在のリソース情報の取得手順について説明する。
【０１１４】
現在の複合機１００のリソース情報の取得においては、システム機器構成に関する情報とメモリ領域等のシステムリソース量を取得する。ステップＳ８０５の処理手順を説明する前にシステム機器構成情報取得について説明する。
【０１１５】
システム機器構成情報は、コントロールサービス層のサービスモジュール（以下、例としてＳＣＳ１２２がシステム機器構成情報を取得する場合について説明する）が取得してシステム機器構成情報構造体に格納する。図１４のフローチャートを参照してＳＣＳ１２２がシステム機器構成情報を取得する場合の手順について説明する。
【０１１６】
複合機１００の主電源が投入されると（ステップＳ１１０１）、ＳＣＳ１２２は複合機１００の各ユニット接続センサ及び各ユニットから機器構成情報の通知を受ける（ステップＳ１１０２）。ＳＣＳ１２２は通知されたシステム機器構成情報をシステム機器構成情報構造体に格納する（ステップＳ１１０３）。システム機器構成情報構造体の構成例を図１５に示す。すなわち、図１５に示す項目の情報をＳＣＳ１２２が取得し、システム機器構成情報構造体に格納する。
【０１１７】
その後、システム機器構成情報構造体を共有メモリ上（物理的にはＲＡＭ上）に配置し（ステップＳ１１０４）、ＳＣＳ１２２に対して登録要求を行ったサービスモジュールやアプリケーションに対して、システム機器構成情報を取得したことを示すシステム機器構成情報通知メッセージを発行する（ステップＳ１１０５）。例えば、ＶＡＳ１４０が起動されるときにシステム機器構成情報通知メッセージがＳＣＳ１２２からＶＡＳ１４０に通知される。
【０１１８】
上記のようにしてシステム機器構成情報構造体が共有メモリ上に配置されることにより他のモジュール（ＶＡＳなど）がシステム機器構成情報構造体にアクセスしてシステム機器構成情報を取得することが可能となる。
【０１１９】
次に、図１６を参照して、図１１のステップＳ８０５におけるＶＡＳ１４０による現在のリソース情報の取得の処理手順について説明する。
【０１２０】
まず、ＶＡＳ１４０はシステム機器構成情報通知メッセージを受信済みか否かをチェックする（ステップＳ１２０１）。受信済みでなければ所定の時間間隔で受信済みか否かのチェックを繰り返す。なお、所定の時間経過してもシステム機器構成情報通知受信を確認できない場合にはタイムアウトとなりシステムエラーとなる。
【０１２１】

システム機器構成情報通知メッセージを受信済みである場合には、システム機器構成情報構造体にアクセスすることによりシステム機器構成情報を取得する（ステップＳ１２０２）。続いて、ｐｒｏｃ構造体にアクセスすることによりシステムリソース量（メモリ領域など）を取得する（ステップＳ１２０３）。

【０１２２】

以上の処理により、使用予定リソースとの比較のために使用する現状のリソース情報を取得することが可能となる。

【０１２３】

（実施の形態３）

実施の形態１、２においては使用予定リソースとの比較のために、メモリ領域などの現在のリソース使用量をｐｒｏｃ構造体から取得していたが、実施の形態３では、現在のリソース使用量を取得せず、予め新規アプリケーション用にリソースを割り当てておく。そして、割り当てたリソース量と、新規アプリケーションが使用しようとするリソース量とを比較して起動の判断を行うようにする。なお、予め割り当てたリソース量は例えばアプリケーション管理ファイルに記録しておく。

【０１２４】

すなわち、例えば、実施の形態１の図８のフローにおいて、現状のリソース使用量を取得する（ステップＳ７０３）代わりに、予め割り当てておいたリソース量を例えばアプリケーション管理ファイルから取得する。そして、そのリソース量とアプリケーションが使用するリソース量とを比較して起動の判断を行う。

【０１２５】

なお、複数の新規アプリケーションが起動される場合には、複数の新規アプリケーションに対してまとめて所定のリソース量を割り当ててもよいし、起動される可能性のある新規アプリケーション個々に対してそれぞれのリソース量を割り当ててもよい。

【０１２６】

前者の場合、まず１つの新規アプリケーションが起動したら、割り当てられたリソース総量から当該新規アプリケーションが使用を予定しているリソース量を減じ、次に起動した新規アプリケーションについては減じた結果のリソース量と使用予定リソース量とを比較して起動に関する判断を行う。これ以降に起動する新規アプリケーションについても同様である。

【０１２７】

また、後者の場合には、例えば新規アプリケーションが使用する予定のリソース量を調べてその量を割り当てる。また、複数の新規アプリケーションに対して等分に割り当ててもよい。

【０１２８】

上記の方法は、メモリ領域のようにアプリケーションが固定的に必要とするリソースに対して有効である。ＣＰＵ使用量等予め割り当てができないリソースについては、判断に用いないか、実施の形態１、２の方法を用いて判断を行う。

【０１２９】

（実施の形態４）

実施の形態１にかかる複合機１００は、ＶＡＳ１４０が全アプリケーションに対して１つのみ存在するものであったが、この実施の形態４にかかる複合機では、各アプリごとに一つのＶＡＳが起動し、各ＶＡＳは対応するアプリに対してのみリソース使用情報取得およびアプリ起動判断を行うものである。また、実施の形態４では実施の形態１の動作を例にとり説明するが、実施の形態２、３についてもＶＡＳの構成を実施の形態４のような構成にすることができる。

【０１３０】

図１７は、実施の形態４にかかる複合機８００の構成を示すブロック図である。図１７に示すように、複合機８００では、複数の仮想アプリケーションサービス（ＶＡＳ）８４１～８４８がアプリケーション１３０の各アプリごとに動作している点が、実施の形態１に

かかる複合機１００と異なっている。

【０１３１】

ＶＡＳ８４１～８４８は、プリンタアプリ１１１、コピーアプリ１１２、ファックスアプリ１１３、スキャナアプリ１１４、ネットファイルアプリ１１５、工程検査アプリ１１６、新規アプリ１１７および１１８に対応して、リソース使用情報取得処理およびアプリ起動判断処理を行うようになっている。なお、図１８に示すように、新規アプリケーションに対してのみＶＡＳを備えるようにしてもよい。

【０１３２】

図１９は、実施の形態４にかかる複合機８００のＶＡＳ８４１～８４８の構成と、ＶＡＳ８４１～８４８と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図１７では、アプリケーション１３０として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ１１７，１１８の例を示し、更にこれら各アプリに対応したＶＡＳ８４１，８４２，８４７および８４８を例として示しているが、他のアプリの場合も同様の構成である。

【０１３３】

また、実施の形態４にかかる複合機８００では、実施の形態１の複合機１００と異なり、図１９に示すように、各ＶＡＳ８４１～８４８と各アプリとの間にはＶＡＳ制御プロセス（デーモン）８０１が動作している。

【０１３４】

このＶＡＳ制御プロセス（デーモン）８０１は、各アプリからアプリ登録要求メッセージを受信してアプリ登録処理を行うとともに、アプリ登録要求を行ったアプリに対応したＶＡＳ８４１～８４８を生成する。また、ＶＡＳ制御プロセス８０１は、ＨＤ２００に格納されたリソース使用情報ファイル２０１を参照して、アプリ登録要求を行ったアプリについて、リソース使用情報が記録されているか否かをチェックすることにより、アプリの起動が初回の起動か２回目以降の起動かを判断する。そして、初回の起動である場合にはアプリに対応するＶＡＳ８４１～８４８に対してアプリＩＤ、アプリのプロセスＩＤとともにリソース使用情報取得処理要求を行い、２回目以降の起動である場合にはアプリに対応するＶＡＳ８４１～８４８に対してアプリＩＤ、アプリのプロセスＩＤとともにアプリ起動判断処理要求を行うようになっている。

【０１３５】

仮想アプリケーションサービス（ＶＡＳ）８４１～８４８のプロセスには、ディスパッチャ１４４と、リソース使用情報取得スレッド１４１と、アプリ起動判断スレッド１４２とが動作している。

【０１３６】

ディスパッチャ１４４は、アプリケーション１３０やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じてリソース使用情報取得スレッド１４１、アプリ起動判断スレッド１４２に処理要求を行うものである。実施の形態２の複合機８００では、ディスパッチャ１４４は、ＶＡＳ制御プロセス８０１から、アプリＩＤ、アプリのプロセスＩＤとともに、リソース使用情報取得処理要求メッセージまたはアプリ起動判断処理要求メッセージを受信するようになっている。ディスパッチャ１４４は、リソース使用情報取得処理要求メッセージを受信したときには、アプリＩＤ、アプリのプロセスＩＤとともに受信したリソース使用情報取得処理要求メッセージをリソース使用情報取得スレッド１４１に送信し、アプリ起動判断要求メッセージを受信したときには、アプリＩＤ、アプリのプロセスＩＤとともに受信したアプリ起動判断要求メッセージをアプリ起動判断スレッド１４２に送信するようになっている。

【０１３７】

リソース使用情報取得スレッド１４１は、ディスパッチャ１４４からのリソース使用情報取得要求メッセージを受信すると、実施の形態１におけるＶＡＳ１４０と同様に、アプリ起動に必要なリソースの情報を取得してリソース使用情報ファイル２０１をハードディスク（ＨＤ）２００に生成する。

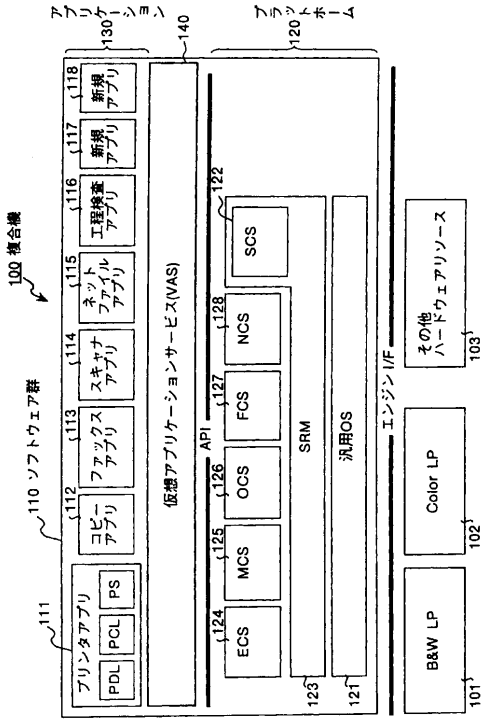
【０１３８】
アプリ起動判断スレッド１４２は、ディスパッチャ１４４からのアプリ起動判断要求メッセージを受信すると、実施の形態１におけるＶＡＳ１４０と同様に、リソース使用情報ファイル２０１を参照して、アプリ起動判断処理を行う。
【０１３９】
実施の形態４の複合機８００におけるＶＡＳ８４１～８４８のリソース使用情報取得スレッド１４１によって実行されるリソース使用情報取得処理、およびアプリ起動判断スレッド１４２によって実行されるアプリ起動判断処理については、実施の形態１の複合機１００におけるＶＡＳ１４０の各スレッドによる処理と同様である。
【０１４０】
このように実施の形態４にかかる複合機８００によれば、実施の形態１にかかる複合機１００と同様に、複合機８００のシステムの安定性を向上させることができる。
【０１４１】
また、実施の形態４にかかる複合機８００では、ＶＡＳ８４１～８４８は起動されるアプリケーション１３０ごとに別個に起動されるので、複数のアプリケーション１３０の起動判断処理を、各アプリケーション１３０に対応するＶＡＳ８４１～８４８で並列に実行することができ、アプリケーションの起動判断処理を効率的に行うことができる。
【０１４２】
なお、実施の形態４にかかる複合機８００では、全てのアプリごとに別個にＶＡＳ８４１～８４８を起動していたが、図１８に示したように一部のアプリに対してのみＶＡＳを起動するように構成しても良い。例えば、新規アプリ１１７，１１８などサードベンダなどの第三者が開発したアプリに対してのみＶＡＳ８４７，８４８を起動してリソース使用情報の取得処理やアプリ起動判断処理を行い、プリンタアプリ１１１やコピーアプリ１１２などの既存のアプリに対してはこのようなサービスを行わないように構成することができる。
【０１４３】
また、実施の形態１および４にかかる複合機１００，８００では、リソースとして、テキストメモリ領域、ヒープ領域、スタック領域、ＣＰＵ占有時間、ＣＰＵ稼働率を利用してリソース使用情報取得およびアプリ起動判断を行っていたが、かかるリソースは一例であり、他のリソースを利用した構成としても良い。
【０１４４】
また、ＶＡＳに構成として上記の構成の他、図２０（ａ）～（ｃ）に示す構成のようにすることもできる。
【０１４５】
図２０（ａ）は、各アプリケーションに対して起動されるＶＡＳを、親ＶＡＳの子プロセスとする場合であり、親ＶＡＳ自体は画面制御権（ユーザインターフェース）を持たない。図２０（ｂ）は、親ＶＡＳ自体は画面制御権（ユーザインターフェース）を持つ場合である。図２０（ｃ）は、各アプリケーションに対応するＶＡＳの機能をスレッドとして提供する場合を示している。
【０１４６】
なお、本発明は、上記の実施の形態に限定されることなく、特許請求の範囲内において、種々変更・応用が可能である。
【０１４７】
【発明の効果】
以上説明したように、本発明によれば、リソースの状況によってアプリケーションの実行ができなくなることを未然に回避して、画像形成装置のシステムの安定性を向上させることができるという効果を奏する。特に、サードベンダなどの第三者が開発した新規アプリケーションが搭載可能な画像形成装置においては、新規アプリケーションがリソースの限界を意識した処理を行っていない場合でも、リソース不足に伴ってシステムが不安定になることを回避でき、画像形成装置のシステムの安定性を向上させることができるという効

果を奏する。
【図面の簡単な説明】
【図１】実施の形態１にかかる複合機の構成を示すブロック図である。
【図２】実施の形態１にかかる複合機の構成の他の例を示すブロック図である。
【図３】実施の形態１にかかる複合機のＶＡＳの構成と、ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの関係を示すブロック図である。
【図４】実施の形態１にかかる複合機におけるリソース使用情報ファイルの内容例を示す説明図である。
【図５】実施の形態１にかかる複合機におけるＶＡＳのリソース使用情報取得スレッドが参照するｐｒｏｃ構造体の一例を示す説明図である。
【図６】実施の形態１にかかる複合機におけるＶＡＳの制御スレッドよるアプリ登録およびアプリ起動回数判断の処理手順を示すフローチャートである。
【図７】実施の形態１にかかる複合機におけるＶＡＳのリソース使用情報取得スレッドによるリソース使用情報取得の処理手順を示すフローチャートである。
【図８】実施の形態１にかかる複合機におけるＶＡＳのアプリ起動判断スレッドによるアプリ起動可否判断の処理手順を示すフローチャートである。
【図９】実施の形態２にかかる複合機におけるアプリケーション管理ファイルの内容例を示す図である。
【図１０】実施の形態２にかかる複合機におけるＮＶ－ＲＡＭの構成例とＨＤの構成例を示す図である。
【図１１】実施の形態２にかかる複合機におけるアプリケーションの起動の際に、ＶＡＳが実行する処理を示すフローチャートである。
【図１２】リソース判定結果通知メッセージを受信したアプリケーションの動作を示すフローチャートである。
【図１３】アプリケーション起動終了手続きを示すフローチャートである。
【図１４】実施の形態２にかかる複合機においてシステム機器構成情報を取得する手順を示すフローチャートである。
【図１５】実施の形態２にかかる複合機におけるシステム機器構成情報構造体の構成例を示す図である。
【図１６】実施の形態２にかかる複合機における現在のリソース情報の取得の処理手順を示すフローチャートである。
【図１７】実施の形態４にかかる複合機の構成を示すブロック図である。
【図１８】実施の形態４にかかる複合機の構成の他の例を示すブロック図である。
【図１９】実施の形態４にかかる複合機のＶＡＳの構成と、ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの関係を示すブロック図である。
【図２０】複合機におけるＶＡＳの構成を示す図である。
【符号の説明】
１００　複合機
１０１　白黒ラインプリンタ
１０２　カラーラインプリンタ
１０３　ハードウェアリソース
１１０　ソフトウェア群
１１１　プリンタアプリ
１１２　コピーアプリ
１１３　ファックスアプリ
１１４　スキャナアプリ
１１５　ネットファイルアプリ
１１６　工程検査アプリ
１１７，１１８　新規アプリ
１２０　プラットホーム

１２１　汎用ＯＳ
１２２　ＳＣＳ
１２３　ＳＲＭ
１２４　ＥＣＳ
１２５　ＭＣＳ
１２６　ＯＣＳ
１２７　ＦＣＳ
１２８　ＮＣＳ
１３０　アプリケーション
１４０，８４１～８４８　仮想アプリケーションサービス（ＶＡＳ）
１４１　リソース使用情報取得スレッド
１４２　アプリ起動判断スレッド
１４３　制御スレッド
１４４　ディスパッチャ
１５０　コントロールサービス層
２００　ハードディスク（ＨＤ）
２０１　リソース使用情報ファイル
２１０　ＲＡＭ
２１１　ｐｒｏｃ構造体
２１２　ｕ領域
８００　複合機
８０１　ＶＡＳ制御プロセス

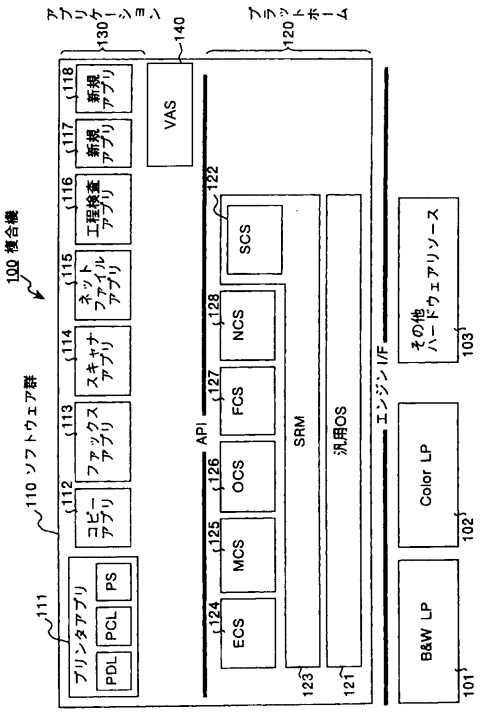
【図１】

実施の形態１にかかる複合機の構成を示すブロック図



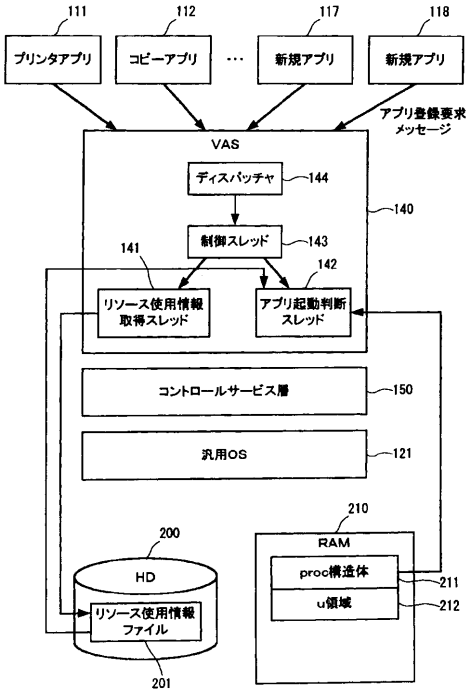
【図２】

実施の形態１にかかる複合機の構成の他の例を示すブロック図



【図３】

実施の形態１にかかる複合機のＶＡＳの構成と、
ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの
関係を示すブロック図である。



【図４】

実施の形態１にかかる複合機における
リソース使用情報ファイルの内容例を示す説明図

リソース使用情報ファイル　201

アプリID	テキストメモリ領域サイズ	ヒープ領域サイズ	スタック領域サイズ	CPU占有時間
101	3MB	1MB	10KB	70ms
102	2MB	500KB	5KB	60ms
103	4MB	200KB	3KB	100ms
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・

【図５】

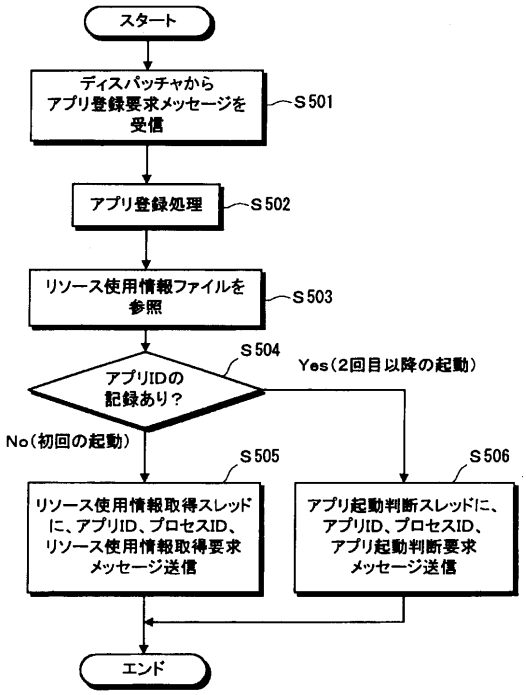
実施の形態１にかかる複合機における
ＶＡＳのリソース使用情報取得スレッドが参照する
ｐｒｏｃ構造体の一例を示す説明図

proc構造体　211

メンバ名	説明
p__pid	プロセスID
p__rtime	CPU占有時間
p__txtsz	テキストメモリ領域サイズ
p__heapsz	ヒープ領域サイズ
p__stacksz	スタック領域サイズ
・	・
・	・
・	・

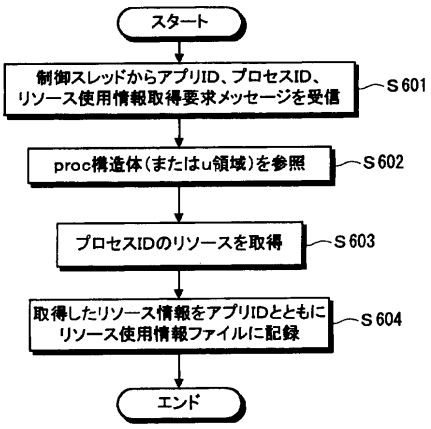
【図６】

実施の形態１にかかる複合機における
ＶＡＳの制御スレッドによるアプリ登録および
アプリ起動回数判断の処理手順を示すフローチャート



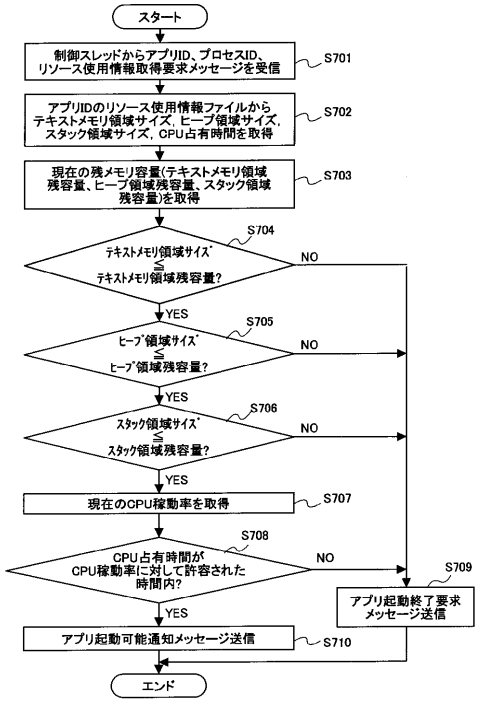
【図７】

実施の形態１にかかる複合機における
ＶＡＳのリソース使用情報取得スレッドによる
リソース使用情報取得の処理手順を示すフローチャート



【図８】

実施の形態１にかかる複合機における
ＶＡＳのアプリ起動判断スレッドによるアプリ起動可否判断の
処理手順を示すフローチャート



【図９】

実施の形態２にかかる複合機における
アプリケーション管理ファイルの内容例を示す図

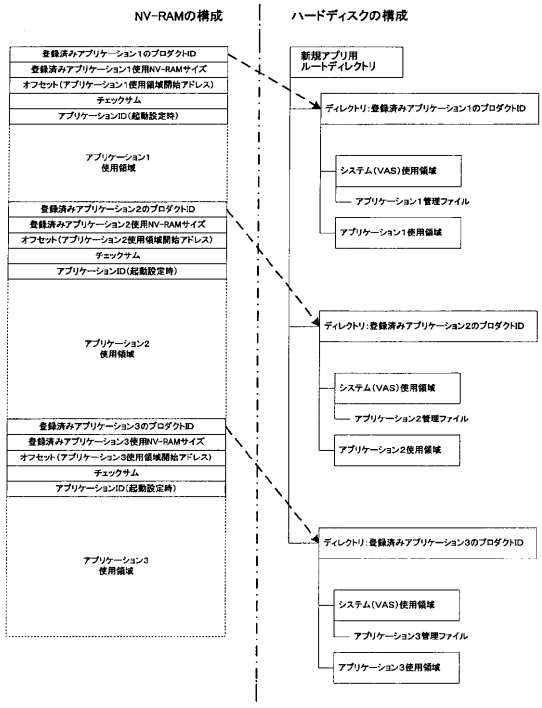
アプリケーション管理ファイル

・アプリケーション名
・アプリケーション　バージョン
・ベンダー名
・連絡先(TEL，FAX，E-MAIL)
・アプリケーション概要
・切り替えキー情報
・実装形態(ICカード、HDD、ネットワーク、・・・)
・起動コマンド(実行ファイル名)

＊使用リソース情報
　メモリ関連
　　・テキストメモリサイズ
　　・ヒープメモリサイズ
　　・スタックサイズ
　システム構成関連
　　・HDDユニット
　　・拡張給紙トレイユニット
　　・大量給紙トレイユニット
　　・タンデムトレイユニット
　　・両面ユニット
　　・フィニッシャ
　　・排紙トレイユニット
　　・原稿処理ユニット(DF)
　　・課金ユニット
　各ユニット詳細情報
　　トレイ関連
　　　・トレイ段数
　　　・トレイタイプ
　　　・紙検知機能有無
　　両面ユニットタイプ
　　フィニッシャ関連
　　　・タイプ
　　　・ステープル
　　　・パンチ
　　　・ソート
　　　・中綴じ
　　拡張排紙トレイ関連
　　　・拡張排紙トレイ段数
　　　・紙検知機能
　　原稿処理ユニット関連
　　　・原稿処理ユニットタイプ
　　　・機能

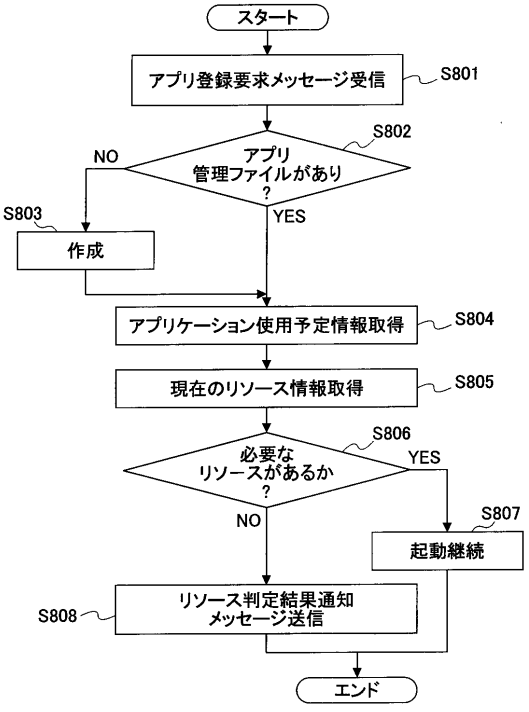
【図１０】

実施の形態２にかかる複合機における
ＮＶ－ＲＡＭの構成例とＨＤの構成例を示す図



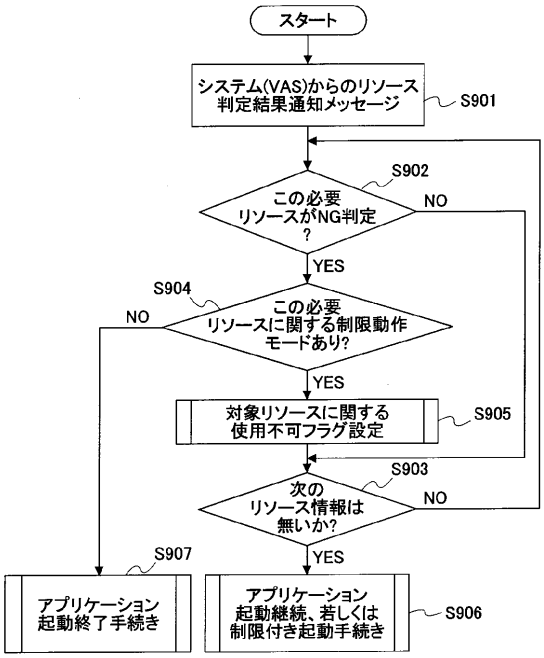
【図１１】

実施の形態２にかかる複合機における
アプリケーションの起動の際に、
ＶＡＳが実行する処理を示すフローチャート



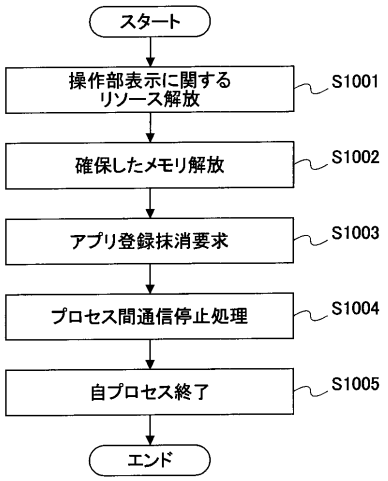
【図１２】

リソース判定結果通知メッセージを受信したアプリケーションの動作を示すフローチャート



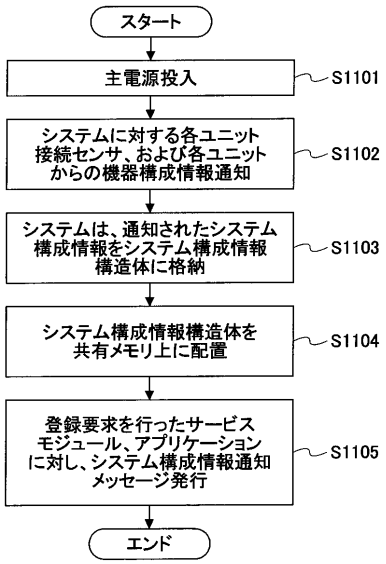
【図１３】

アプリケーション起動終了手続きを示すフローチャート



【図１４】

実施の形態２にかかる複合機において
システム機器構成情報を取得する手順を示すフローチャート



【図１５】

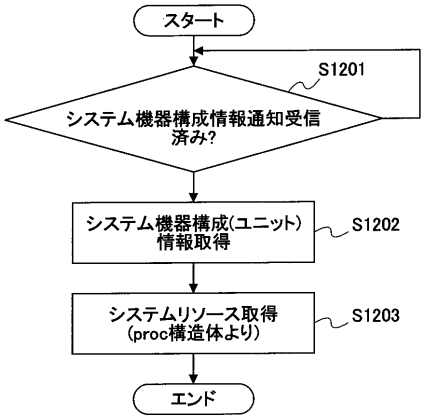
実施の形態２にかかる複合機における
システム機器構成情報構造体の構成例を示す図

システム機器構成情報 構造体

メンバ名	説明
	ユニット接続情報
hdd_unit	HDDユニット接続フラグ
ext_feed_tray	拡張給紙トレイユニット接続フラグ
large_tray	大量給紙トレイユニット接続フラグ
tandem_tray	タンデムトレイユニット接続フラグ
dpx_unit	両面ユニット接続フラグ
fini	フィニッシャ接続フラグ
exit_tray	排紙トレイユニット接続フラグ
df_unit	原稿処理ユニット(DF)接続フラグ
charge_unit	課金ユニット接続フラグ
	各ユニット詳細情報
	トレイ関連
tray_num	トレイ段数
tray_type	トレイタイプ
tray_sense	紙検知機能有無
dpx_type	両面ユニットタイプ
	フィニッシャ関連
fini_type	タイプ
staple	ステープル
punch	パンチ
	ソート
	中綴じ
	拡張排紙トレイ関連
exit_tray_num	拡張排紙トレイ段数
exit_tray_sense	紙検知機能
	原稿処理ユニット関連
df_type	原稿処理ユニットタイプ
df_func	機能
・	・
・	・
・	・

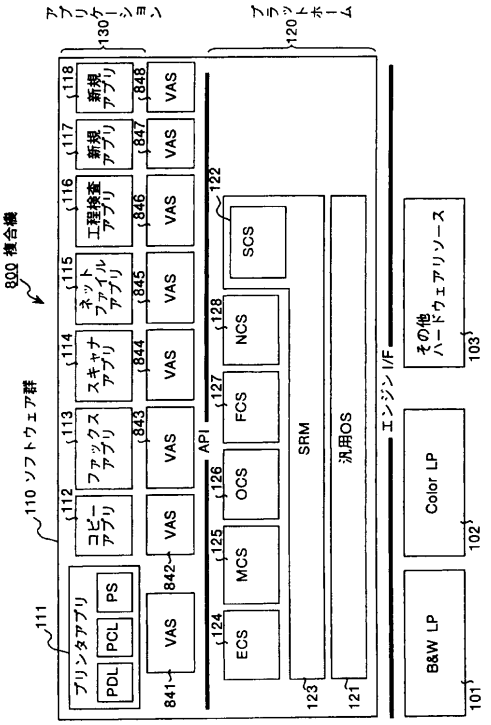
【図１６】

実施の形態２にかかる複合機における
現在のリソース情報の取得の処理手順を示すフローチャート



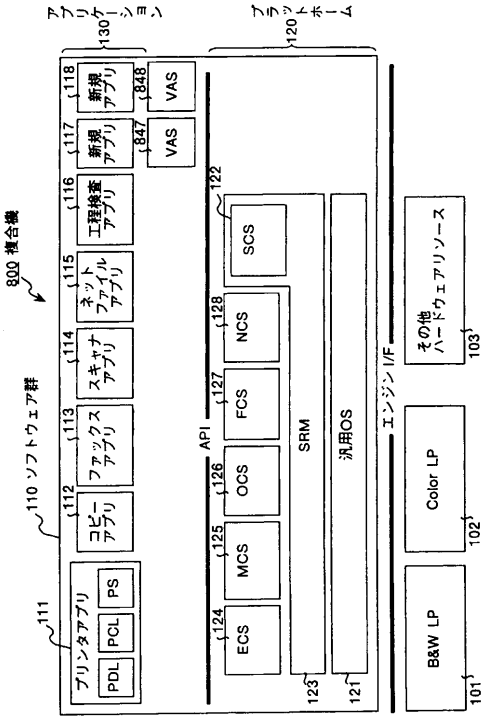
【図１７】

実施の形態４にかかる複合機の構成を示すブロック図



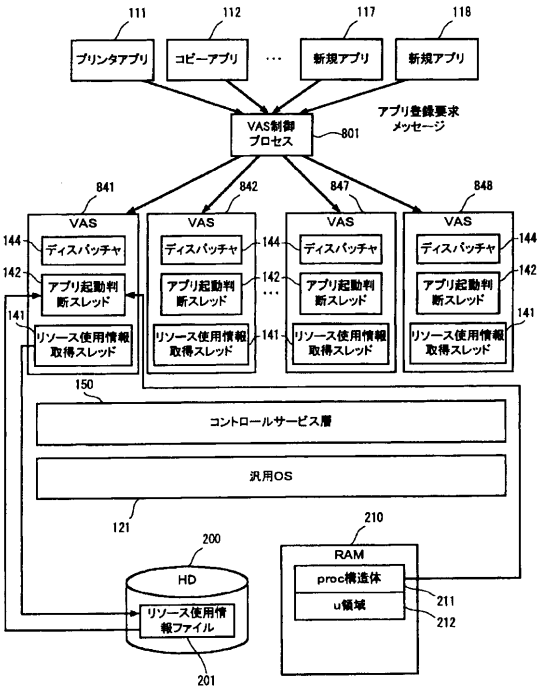
【図１８】

実施の形態４にかかる複合機の構成の他の例を示すブロック図



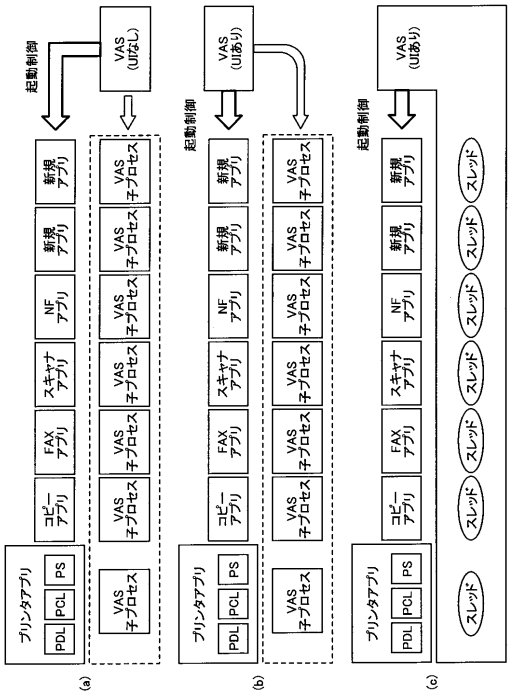
【図１９】

実施の形態４にかかる複合機のＶＡＳの構成と、
ＶＡＳと各アプリ、コントロールサービス層
および汎用ＯＳとの関係を示すブロック図



【図２０】

複合機におけるＶＡＳの構成を示す図



フロントページの続き

Ｆターム(参考) 5B076 AB18 AB20 BB06
5C062 AA02 AA05 AB42 AC55 AC58 BA04