(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2018-124717
(P2018-124717A)
(43)公開日　平成30年8月9日(2018.8.9)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 12/00	(2006.01)	G06F 12/00	501B	
		G06F 12/00	531Z	

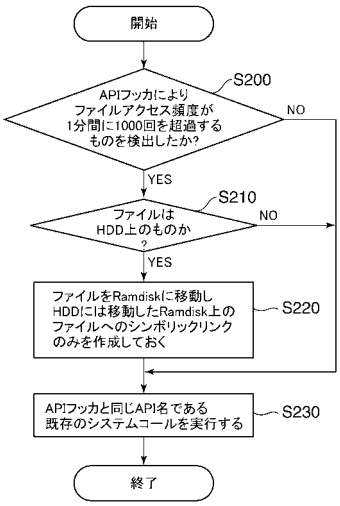
審査請求 未請求　請求項の数 16　ＯＬ　（全 19 頁）

(21)出願番号　特願2017-15414(P2017-15414)
(22)出願日　平成29年1月31日(2017.1.31)

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100125254
弁理士　別役　重尚
(72)発明者　宮田　孝明
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

(54)【発明の名称】情報処理装置及びその制御方法、並びにプログラム

(57)【要約】
【課題】ＨＤＤに対する読み書きの失敗や、物理故障を抑止することが可能となる情報処理装置及びその制御方法、並びにプログラムを提供する。
【解決手段】ＭＦＰ８において、ファイルディスクリプタを扱うシステムコールへのフックを行うＡＰＩフッカを用いて、ＨＤＤ１５へのアクセス頻度がファイル毎に記録される。また、この記録されたアクセス頻度が一定量を超過したファイルに対して、ＡＰＩフッカを用いて、そのファイルのアクセス先がＨＤＤ１５からＲＡＭ２４のＲａｍｄｉｓｋ上に移動する。
【選択図】図２



【特許請求の範囲】
【請求項１】
不揮発性記憶媒体と、揮発性記憶媒体とを備えた情報処理装置において、
ファイルディスクリプタを扱うシステムコールのフックを行うフック手段と、
前記フック手段を用いて、前記不揮発性記憶媒体へのアクセス頻度をファイル毎に記録する記録手段と、
前記フック手段を用いて、前記記録されたアクセス頻度が一定量を超過したファイルに対するアクセス先を前記不揮発性記憶媒体から前記揮発性記憶媒体に移動する移動手段とを備えることを特徴とする情報処理装置。
【請求項２】
前記ファイルへの前記記録されたアクセス頻度が前記一定量を超過した後に前記一定量を下回った場合、前記フック手段を用いて前記ファイルのアクセス先を前記不揮発性記憶媒体に戻す第１の再移動手段を更に備えることを特徴とする請求項１記載の情報処理装置。
【請求項３】
シャットダウンを開始した時に、前記フック手段を用いて前記ファイルのアクセス先を前記不揮発性記憶媒体に戻す第２の再移動手段を更に備えることを特徴とする請求項２記載の情報処理装置。
【請求項４】
前記ファイルへの前記記録されたアクセス頻度が一定量を超過した場合、前記移動手段は、前記フック手段を用いて前記ファイルを前記揮発性記憶媒体に移動し、前記フック手段を用いて前記不揮発性記憶媒体には前記揮発性記憶媒体の前記ファイルへのシンボリックリンクのみを残すことを特徴とする請求項３記載の情報処理装置。
【請求項５】
前記第１及び第２の再移動手段は、前記フック手段を用いて、前記不揮発性記憶媒体の上の前記シンボリックリンクを削除すると共に、前記揮発性記憶媒体の上に移動された前記ファイルを前記不揮発性記憶媒体に戻すことを特徴とする請求項４記載の情報処理装置。
【請求項６】
前記ファイルへの前記記録されたアクセス頻度が一定量を超過した場合、前記移動手段は、前記フック手段を用いて前記ファイルを前記揮発性記憶媒体の上に移動すると共に、前記ファイルのアクセス先を前記揮発性記憶媒体に変更することを特徴とする請求項３記載の情報処理装置。
【請求項７】
前記第１及び第２の再移動手段は、前記フック手段を用いて、前記揮発性記憶媒体の上に移動された前記ファイルを前記不揮発性記憶媒体に戻し、その後、前記アクセス先の変更をさせなくすることを特徴とする請求項６記載の情報処理装置。
【請求項８】
不揮発性記憶媒体と、揮発性記憶媒体とを備えた情報処理装置において、
ファイルディスクリプタを扱うシステムコールのフックを行うフック手段と、
前記フック手段を用いて、前記不揮発性記憶媒体へのアクセス頻度をファイル毎に記録する記録手段と、
前記記録されたアクセス頻度が一定量を超過したファイルに対するアクセスが前記ファイルへの追記である場合、前記フック手段を用いて、前記ファイルのうち前記追記されたデータのアクセス先のみを前記揮発性記憶媒体に移動する移動手段とを備えることを特徴とする情報処理装置。
【請求項９】
前記ファイルへの前記記録されたアクセス頻度が一定量を超過した場合であって、前記ファイルに対するアクセスが前記ファイルへの追記でない場合、前記フック手段を用いて前記ファイルを一定サイズのデータに分割する分割手段をさらに備え、

前記移動手段は、前記分割手段により前記ファイルが分割された場合、前記分割されたデータのうち前記ファイルに対するアクセスがあった箇所のデータのみを前記フック手段を用いて前記揮発性記憶媒体の上に移動することを特徴とする請求項８記載の情報処理装置。

【請求項１０】

前記ファイルへの前記記録されたアクセス頻度が一定量を超過した場合であって、前記ファイルのサイズが前記揮発性記憶媒体の容量の特定の割合未満である場合、前記移動手段は前記フック手段を用いて前記ファイルの全体を前記揮発性記憶媒体に移動することを特徴とする請求項９記載の情報処理装置。

【請求項１１】

前記移動手段による移動が行われた場合、前記移動手段により前記ファイルはその全体、前記追記されたデータ、及び前記分割されたデータのいずれが前記揮発性記憶媒体に移動されているかの区別を管理すると共に、前記追記されたデータ及び前記分割されたデータのいずれかのデータが前記揮発性記憶媒体に移動されている場合、前記ファイルと前記移動されたデータとのオフセットの対応を管理する管理手段を更に備え、

前記ファイルへの前記記録されたアクセス頻度が前記一定量を超過した後に前記一定量を下回った場合、前記管理される区別に基づき、前記移動手段により前記ファイルはその全体、前記追記されたデータ、及び前記分割されたデータのいずれが前記揮発性記憶媒体に移動されているかを判定する第１の判定手段と、

前記第１の判定手段による判定の結果、前記追記されたデータ及び前記分割されたデータのいずれかのデータが前記揮発性記憶媒体に移動されている場合、前記フック手段を用いて、前記不揮発性記憶媒体の上の前記ファイルを前記移動されたデータにより更新すると共に前記揮発性記憶媒体に移動されたデータを削除することを特徴とする請求項１０記載の情報処理装置。

【請求項１２】

シャットダウンを開始したときに、前記移動手段により移動された前記追記されたデータ及び前記分割されたデータのいずれかのデータが前記揮発性記憶媒体に存在する場合、前記フック手段を用いて、前記不揮発性記憶媒体の上の前記ファイルを前記移動されたデータにより更新すると共に前記揮発性記憶媒体に移動されたデータを削除することを特徴とする請求項９乃至１１のいずれか１項に記載の情報処理装置。

【請求項１３】

前記フック手段はＡＰＩフッカであることを特徴とする請求項１乃至１２のいずれか１項に記載の情報処理装置。

【請求項１４】

不揮発性記憶媒体と、揮発性記憶媒体とを備えた情報処理装置の制御方法において、

ファイルディスクリプタを扱うシステムコールのフックをＡＰＩフッカで行うフックステップと、

前記ＡＰＩフッカを用いて、前記不揮発性記憶媒体へのアクセス頻度をファイル毎に記録する記録手段と、

前記ＡＰＩフッカを用いて、前記記録されたアクセス頻度が一定量を超過したファイルに対するアクセス先を前記不揮発性記憶媒体から前記揮発性記憶媒体に移動する移動ステップとを有することを特徴とする制御方法。

【請求項１５】

不揮発性記憶媒体と、揮発性記憶媒体とを備えた情報処理装置の制御方法において、

ファイルディスクリプタを扱うシステムコールのフックをＡＰＩフッカで行うフックステップと、

前記ＡＰＩフッカを用いて、前記不揮発性記憶媒体へのアクセス頻度をファイル毎に記録する記録ステップと、

前記記録されたアクセス頻度が一定量を超過したファイルに対するアクセスが前記ファイルへの追記である場合、前記ＡＰＩフッカを用いて、前記ファイルのうち前記追記され

たデータのアクセス先のみを前記揮発性記憶媒体に移動する移動ステップとを有することを特徴とする制御方法。
【請求項１６】
請求項１４又は請求項１５に記載の制御方法を実行することを特徴とするプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、情報処理装置及びその制御方法、並びにプログラムに関し、特に、搭載されるＨＤＤのアクセス制御を行う情報処理装置及びその制御方法、並びにプログラムに関する。
【背景技術】
【０００２】
従来より、画像形成装置は、ネットワーク送受信可能な状態での待機時（ネットワーク待機時）や節電状態での待機時（節電待機時）に、内部のアプリケーションやネットワーク先のサーバから頻繁に搭載されるＨＤＤへのアクセスを受ける場合がある。
【０００３】
かかる場合、ＨＤＤ稼働時間における、ＨＤＤの磁気ヘッドがプラッタ（磁気ディスク）上にいる時間（アクセス時間）が、コンシューマ向けＨＤＤの安定動作の目安となる、ＨＤＤ　ｄｕｔｙ　２０％を超える可能性がある。このように、アクセス時間がＨＤＤ　ｄｕｔｙ　２０％を超えると、プラッタの回転により軸受けから気化したグリスが、磁気ヘッドとプラッタ間の気圧変化により、磁気ヘッドへゴミ（パーティクル）として付着する確率が上がる。また、その結果、ＨＤＤ読み書きに失敗する確率が上がるだけでなく、最悪の場合はＨＤＤが物理的に故障する恐れもある。
【０００４】
ここで、ＨＤＤ　Ｄｕｔｙの比率を低減するには、ＨＤＤ上のデータを、一時的に異なる記憶装置に退避し、その記憶装置上でデータのアクセスを行う方法がある。
【０００５】
例えば、省電力機能を有する画像形成装置において、サスペンド時にＨＤＤのデータをＲＡＭ上のＲａｍｄｉｓｋに移動し、レジューム時にそのＲａｍｄｉｓｋに移動したデータをＨＤＤに戻すという技術が知られている（例えば特許文献１参照）。これにより、省電力状態時にＨＤＤ上のデータにアクセスする必要がなくなるため、ＨＤＤの電源をＯＦＦすることが可能となる。
【先行技術文献】
【特許文献】
【０００６】
【特許文献１】特許第５２０９９９３号公報
【発明の概要】
【発明が解決しようとする課題】
【０００７】
しかしながら、特許文献１では、ＨＤＤ上のデータのＲａｍｄｉｓｋへの移動を、サスペンド時に限定している。すなわち、この技術では、ネットワーク待機時等、節電待機時以外のＨＤＤのアクセス頻度が多いタイミングにおけるＨＤＤ　Ｄｕｔｙの比率を低減することはできない。
【０００８】
本発明は、ＨＤＤに対する読み書きの失敗や、物理故障を抑止することが可能となる情報処理装置及びその制御方法、並びにプログラムを提供することである。
【課題を解決するための手段】
【０００９】
本発明の請求項１に係る情報処理装置は、不揮発性記憶媒体と、揮発性記憶媒体とを備えた情報処理装置において、ファイルディスクリプタを扱うシステムコールのフックを行

うフック手段と、前記フック手段を用いて、前記不揮発性記憶媒体へのアクセス頻度をファイル毎に記録する記録手段と、前記フック手段を用いて、前記記録されたアクセス頻度が一定量を超過したファイルに対するアクセス先を前記不揮発性記憶媒体から前記揮発性記憶媒体に移動する移動手段とを備えることを特徴とする。

【００１０】

本発明の請求項８に係る情報処理装置は、不揮発性記憶媒体と、揮発性記憶媒体とを備えた情報処理装置において、ファイルディスクリプタを扱うシステムコールのフックを行うフック手段と、前記フック手段を用いて、前記不揮発性記憶媒体へのアクセス頻度をファイル毎に記録する記録手段と、前記記録されたアクセス頻度が一定量を超過したファイルに対するアクセスが前記ファイルへの追記である場合、前記フック手段を用いて、前記ファイルのうち前記追記されたデータのアクセス先のみを前記揮発性記憶媒体に移動する移動手段とを備えることを特徴とする。

【発明の効果】

【００１１】

本発明によれば、ＨＤＤに対する読み書きの失敗や、物理故障を抑止することが可能となる。

【図面の簡単な説明】

【００１２】

【図１】本発明の情報処理装置としてのＭＦＰの詳細なハードウェア構成を示すブロック図である。

【図２】実施例１における、ＨＤＤ上のファイルの移動処理の手順を示すフローチャートである。

【図３】実施例１における、ファイル再移動処理の手順を示すフローチャートである。

【図４】実施例１における、シャットダウン開始時のファイル再移動処理の手順を示すフローチャートである

【図５】実施例２における、ＨＤＤ上のファイルの移動処理の手順を示すフローチャートである。

【図６】実施例２における、ファイル再移動処理の手順を示すフローチャートである。

【図７】実施例２における、シャットダウン開始時のファイル再移動処理の手順を示すフローチャートである。

【図８】実施例３における、ＨＤＤ上のファイルの移動処理の手順を示すフローチャートである。

【図９】実施例３における、ファイル再移動処理の手順を示すフローチャートである。

【図１０】実施例３における、シャットダウン開始時のファイル再移動処理の手順を示すフローチャートである。

【図１１】実施例１における、アプリケーションからのＨＤＤ上のファイルに対するアクセスが１分間に１０００回未満の場合の、ＡＰＩフッカを用いた処理の流れを示す図である。

【図１２】実施例１における、アプリケーションからのＨＤＤ上のファイルに対するアクセスが１分間に１０００回以上となった場合の、ＡＰＩフッカを用いた処理の流れを示す図である。

【発明を実施するための形態】

【００１３】

以下、本発明を実施するための形態について図面を用いて説明する。

【００１４】

図１は、本発明の情報処理装置としてのＭＦＰ８の詳細なハードウェア構成を示すブロック図である。

【００１５】

操作部４は、ＬＣＤ／タッチパネル１０と操作キー１１から構成される。ＬＣＤ／タッチパネル１０は、ユーザへの情報の表示や、ボタン画像を表示した後、ボタンを指等で押

すことでインタラクティブな操作を可能とすることを目的としている。操作キー１１は、印刷部数等を指定するための数字ボタン、コピーボタン、ストップボタン等の良く使用するためのボタンを物理的なボタンスイッチ等で構成する。

【００１６】

人感センサ４１は操作部４の近傍に物理的に配置されているものである。人感センサ４１の検知閾値は、操作部４にて段階的に設定可能であり、例えば、ユーザがＭＦＰ８に近づいてきたとき、まだ遠い時点でも検知するように設定できる。また、ごく近くまでユーザが近づかないと検知しないようにも設定できる。ＭＦＰ８では、人感センサ４１がユーザを検知したときに、電力モードが節電モードであった場合、スタンバイモードに遷移させることも可能である。ここで、節電モードとは、ＣＰＵ１６がＳ３状態であることを含む電力モードであり、スタンバイモードとは、ＣＰＵ１６がＳ０状態であることを含む電力モードである。

【００１７】

節電ボタン１２は操作部４の近傍、もしくは同一ユニット上に物理的に配置されている、節電モードから復帰するためのスイッチである。つまり、後述する電源制御部１８のスイッチ２２により操作部４の電源が落とされている場合でも節電ボタン１２に対する押下を検出することが可能なように、操作部４とは電気的に分離された構成になっている。すなわち、節電ボタン１２は、操作部４とは別個に独立した状態で制御部５と接続している。この節電ボタン１２をユーザが押下することで、ＭＦＰ８を節電モードに入れることや、スタンバイモードにトグル遷移させることが可能である。

【００１８】

ユーザ認証入力装置９は、認証プリントのためにユーザの認証を行うユーザ認証部６を有する。

【００１９】

コントローラ部１は、制御部５、ネットワーク接続部１３、及び印刷データ記憶部１５から構成される。ネットワーク接続部１３は、ＰＣ３９から例えばネットワーク４０を介して依頼を受け付ける。印刷データ記憶部１５は、受信した印刷データを蓄積するＨＤＤからなるメモリである（以下、「ＨＤＤ１５」という。）。制御部５は、ＣＰＵ１６、ＳＰＩ　Ｆｌａｓｈ２３、及びＲＡＭ２４から構成されるデバイスであり、ネットワーク接続部１３及びＨＤＤ１５と相互に接続され、コントローラ部１全体を制御する。ＣＰＵ１６は、ネットワーク接続部１３を介して受信した外部データが印刷データか否かの判断等の処理を行う。ＳＰＩ　Ｆｌａｓｈ２３は、ＣＰＵ１６の起動に必要な制御用ファームウェアを格納する。ＲＡＭ２４は、ＣＰＵ１６への命令を一時的に格納すると共に、必要に応じてそのメモリ領域の一部をデータ書き込みが可能なＲａｍｄｉｓｋにする。

【００２０】

また、プリンタ部３、スキャナ部２、ユーザ認証入力装置９、操作部４の電源を細かく制御して、節電モードからのユーザの待ち時間を減らすため、電源制御部１８はＣＰＵ１６がリモート制御可能なスイッチ１９～２２を有する。これらのスイッチ１９～２２は図１に示す矢印のように、プリンタ部３、スキャナ部２、ユーザ認証入力装置９、及び操作部４のそれぞれに接続され、ＣＰＵ１６はこれらのスイッチ１９～２２のオン・オフを制御することでＭＦＰ８全体の電力を制御する。尚、これらのスイッチ１９～２２はＦＥＴやリレーなどで構成してもよい。

【００２１】

さらにコントローラ部１内のＨＤＤ１５は、スタンバイモードであっても、ＣＰＵ１６の指示によりＨＤＤ１５単体のみを省電力状態に移動させることが可能である。

【００２２】

（実施例１）

次に図２～４，１１，１２を用いて、実施例１に係るファイルディスクリプタを扱うシステムコールをフックするＡＰＩフッカの動作を説明する。

【００２３】

図２は、実施例１における、ＨＤＤ１５上のファイルの移動処理の手順を示すフローチャートである。この処理では、ＨＤＤ１５上のファイルに対して１分間に１０００回以上のアクセスがあった場合、ＨＤＤ１５にそのファイルのシンボリックリンクが作成され、Ｒａｍｄｉｓｋにそのファイルが移動する。尚、この処理は、ＣＰＵ１６が、ＨＤＤ１５上にある本処理を実行するためのプログラムをＲＡＭ２４上で展開することにより実行される。

【００２４】

ここで、ＡＰＩフッカとは、既存のシステムコールと同じＡＰＩ名でありながら、既存のシステムコールの動作と、これに先立ち、プログラマの所望の動作を付加するライブラリ関数のことである。図２の場合では、ＡＰＩフッカを用いて、既存のｒｅａｄ（），ｗｒｉｔｅ（）などのファイルディスクリプタを扱うシステムコールの実行に先立ち行われる動作として、以下の３つの動作が付加される。具体的には、ファイルのアクセス頻度の検出と、シンボリックリンクの作成と、Ｒａｍｄｉｓｋへのファイル移動という動作が付加される。このとき、アクセス頻度の情報は、ファイルディスクリプタを扱う各ＡＰＩフッカで共有しておくことで、あるファイルに対するアクセス頻度を計測することが可能となる。

【００２５】

ここで、ＡＰＩフッカと既存のシステムコールが同じＡＰＩ名であるという問題が生じる。この問題は、ＯＳの一つであるＬｉｎｕｘ（登録商標）の場合、同じＡＰＩ名が複数ある場合、ロードされたライブラリ順にＡＰＩ名を検索し、最も早くに検索できたＡＰＩ名が利用されるという特徴を用いて解決する。具体的には、環境変数ＬＤ＿ＰＲＥＬＯＡＤを設定することで、ＡＰＩフッカを含む関数ライブラリが最も早くロードされるようにしておくようにする。これにより、アプリケーションが例えばｒｅａｄ（）システムコールを呼び出した際であっても、既存のシステムコールであるｒｅａｄ（）ではなく、ＡＰＩフッカであるｒｅａｄ（）が呼び出されることになる。

【００２６】

ＣＰＵ１６は、ステップＳ２００にて、ＡＰＩフッカにより、アクセス頻度が１分間に１０００回を超過するファイルの検出を行う。ここで、ファイルが検出されない場合、ステップＳ２３０の処理に移る。尚、本実施例ではアクセス頻度が多いファイルか否かの判断基準として、１分間に１０００回という基準を用いたが、これに限定されるものではなく、アクセス頻度が一定量を超えるものであればよい。以下後述する他の実施例においても同様である。

【００２７】

ステップＳ２００にて、ファイルが検出された場合は、ステップＳ２１０に移り、ファイルがＨＤＤ１５上のものか否かを判定する。ここで、ＨＤＤ１５上のものでなければ、ステップＳ２３０の処理に移る。

【００２８】

ステップＳ２１０にて、ファイルがＨＤＤ１５上のものであれば、ステップＳ２２０に移り、ＡＰＩフッカにより、ＨＤＤ１５にそのファイルのシンボリックリンクのみ作成するとともに、そのファイルをＲａｍｄｉｓｋに移動する。その後ステップＳ２３０の処理に移る。

【００２９】

ステップＳ２３０では、ＡＰＩフッカと同じＡＰＩ名である既存のシステムコールを実行し、本処理を終了する。

【００３０】

これらの処理により、ＨＤＤ１５上にあるファイルへのアクセス頻度が、１分間に１０００回を超過した場合、アクセス先がＲａｍｄｉｓｋに変更される。これにより、ＨＤＤ１５への大量のアクセスを抑止することができ、ＨＤＤ　Ｄｕｔｙ比率を低減することが可能となる。

【００３１】

図３は、実施例１における、ファイル再移動処理の手順を示すフローチャートである。この処理では、図２の処理によりＲａｍｄｉｓｋに移動したファイルへのアクセスが、１分間に１０００回のアクセスを下回った場合、そのファイルがＨＤＤ１５上に戻される。尚、この処理は、ＣＰＵ１６が、ＨＤＤ１５上にある本処理を実行するためのプログラムをＲＡＭ２４上で展開することにより実行される。

【００３２】

図３では、ＡＰＩフッカを用いて、ファイルディスクリプタを扱う既存のシステムコールの実行に先立ち行われる動作として、ファイルのアクセス頻度の検出と、シンボリックリンクの削除と、ＨＤＤ１５へのファイル移動という動作が付加される。その他の、ＡＰＩフッカの特徴については、図２の場合と同様である。

【００３３】

ＣＰＵ１６は、ステップＳ３００にて、ＡＰＩフッカにより、アクセス頻度が１分間に１０００回を超過していたものが超過しなくなったファイルの検出を行う。ここで、ファイルが検出されない場合、ステップＳ２３０の処理に移る。

【００３４】

ステップＳ３００にて、ファイルが検出された場合は、ステップＳ３１０に移り、ＡＰＩフッカにより、ＨＤＤ１５上のシンボリックリンクを削除し、Ｒａｍｄｉｓｋ上のその検出されたファイルをＨＤＤ１５に戻す。引き続き、ステップＳ２３０の処理に移る。

【００３５】

ステップＳ２３０では、ＡＰＩフッカと同じＡＰＩ名である既存のシステムコールを実行し、本処理を終了する。

【００３６】

これらの処理により、図２の処理によりＲａｍｄｉｓｋ上に移動したファイルへのアクセス頻度が、１分間に１０００回を下回った場合、そのファイルをＨＤＤ１５に戻すことで、容量に制限の多いＲａｍｄｉｓｋを有効に利用することが可能となる。

【００３７】

図４は、実施例１における、シャットダウン開始時のファイル再移動処理の手順を示すフローチャートである。この処理では、図２の処理によりＲａｍｄｉｓｋに移動した全てのファイルは、シャットダウン開始時にＨＤＤ１５上に戻される。尚、この処理は、ＣＰＵ１６が、ＨＤＤ１５上にある本処理を実行するためのプログラムをＲＡＭ２４上で展開することにより実行される。

【００３８】

図４では、ＡＰＩフッカを用いて、ファイルディスクリプタを扱う既存のシステムコールの実行に先立ち行われる動作として、以下の４つの動作が付加される。具体的には、シャットダウン開始の検出と、Ｒａｍｄｉｓｋに移動したファイルの検出と、シンボリックリンクの削除と、ＨＤＤ１５へのファイル移動という動作が付加される。その他の、ＡＰＩフッカの特徴については、図２の場合と同様である。

【００３９】

ＣＰＵ１６は、ステップＳ４００にてＡＰＩフッカによりシャットダウンが開始されたかを検出する。シャットダウンが開始されていない場合は、そのまま本処理を終了する。

【００４０】

ステップＳ４００で、シャットダウンの開始が検出された場合は、ステップＳ４１０に移り、図２の処理によりＲａｍｄｉｓｋ上に移動したファイルが存在するかをＡＰＩフッカにより検出する。ここで、ファイルが検出されない場合は、そのまま本処理を終了する。

【００４１】

ステップＳ４１０で、Ｒａｍｄｉｓｋに移動したファイルが検出された場合は、ステップＳ３１０に移り、ＡＰＩフッカにより、ＨＤＤ１５上のシンボリックリンクを削除すると共に、Ｒａｍｄｉｓｋ上のその検出されたファイルをＨＤＤ１５に戻す。その後、ステップＳ４００の処理に戻る。

【００４２】
これらの処理により、シャットダウン開始時に、図２の処理によりＲａｍｄｉｓｋ上に移動した全てのファイルをＨＤＤ１５に戻すことで、Ｒａｍｄｉｓｋ上のファイルがシャットダウンによって揮発しないようにすることが可能となる。
【００４３】
図１１は、実施例１における、アプリケーションからのＨＤＤ１５上のファイルに対するアクセスが１分間に１０００回未満の場合の、ＡＰＩフッカを用いた処理の流れを示す図である。
【００４４】
図１１では、アプリケーションが、ファイルディスクリプタの関数として、ｗｒｉｔｅ（）関数のみを実行する場合について説明するが、他のファイルディスクリプタの関数が実行された場合も以下の処理と同様の処理が行われる。また、ファイル毎のアクセス頻度の情報は、ファイルディスクリプタを扱う各ＡＰＩフッカで共有される。
【００４５】
まず、アプリケーションにてＨＤＤ１５上のファイル／ＡＰＬ＿ＤＡＴＡ／ｆｉｌｅ＿Ａに対してｗｒｉｔｅ（）関数１１００が実行された場合、ＣＰＵ１６は、ｗｒｉｔｅシステムコール１１２０ではなく、ｗｒｉｔｅ　ＡＰＩフッカ１１１０を呼び出す。
【００４６】
次に、ＣＰＵ１６は、ｗｒｉｔｅ　ＡＰＩフッカ１１１０を用いて、アプリケーションのｗｒｉｔｅ（）関数１１００の引数で指定されたファイル毎に、単位時間当たりのファイルアクセス頻度を計測する。すなわち、この場合はファイル／ＡＰＬ＿ＤＡＴＡ／ｆｉｌｅ＿Ａに対する単位時間当たりのファイルアクセス頻度が計測される。図１１の場合は、このファイル／ＡＰＬ＿ＤＡＴＡ／ｆｉｌｅ＿に対して１分間に１０００回未満のアクセスしかないので、ｗｒｉｔｅシステムコール１１２０がそのまま呼び出される。
【００４７】
最後に、ｗｒｉｔｅシステムコール１１２０により、ＨＤＤ１５上のファイル／ＡＰＬ＿ＤＡＴＡ／ｆｉｌｅ＿Ａにデータがｗｒｉｔｅされる。
【００４８】
図１２は、本実施例における、アプリケーションからのＨＤＤ１５上のファイルに対するアクセスが１分間に１０００回以上となった場合の、ＡＰＩフッカを用いた処理の流れを示す図である。
【００４９】
図１２においても、図１１と同様、アプリケーションが、ファイルディスクリプタの関数として、ｗｒｉｔｅ（）関数のみを実行する場合について説明するが、他のファイルディスクリプタの関数が実行された場合も以下の処理と同様の処理が行われる。また、ファイル毎のアクセス頻度の情報は、ファイルディスクリプタを扱う各ＡＰＩフッカで共有される。
【００５０】
まず、アプリケーションにてＨＤＤ１５上のファイル／ＡＰＬ＿ＤＡＴＡ／ｆｉｌｅ＿Ａに対してｗｒｉｔｅ（）関数１１００が実行された場合、ｗｒｉｔｅ　ＡＰＩフッカ１１１０が呼び出される。
【００５１】
次に、ＣＰＵ１６は、ｗｒｉｔｅ　ＡＰＩフッカ１１１０を用いて、ファイル／ＡＰＬ＿ＤＡＴＡ／ｆｉｌｅ＿Ａに対する単位時間当たりのファイルアクセス頻度を計測する。図１２では、この計測の結果、１分間に１０００回以上のアクセスが発生している。よって、まず、ｗｒｉｔｅ　ＡＰＩフッカ１１１０を用いて、ＨＤＤ１５上のファイル／ＡＰＬ＿ＤＡＴＡ／ｆｉｌｅ＿Ａを、Ｒａｍｄｉｓｋ上の／ｍｎｔ／ｒａｍ／ｆｉｌｅ＿Ａに移動する。次に、ｗｒｉｔｅ　ＡＰＩフッカ１１１０を用いて、ＨＤＤ１５上に、／ｍｎｔ／ｒａｍ／ｆｉｌｅ＿Ａへのシンボリックリンクである／ＡＰＬ＿ＤＡＴＡ／ｆｉｌｅ＿Ａを作成する。その後、ｗｒｉｔｅシステムコール１１２０が呼び出される。

【００５２】
最後に、ｗｒｉｔｅシステムコール１１２０により、ＨＤＤ１５上のシンボリックリンク／ＡＰＬ＿ＤＡＴＡ／ｆｉｌｅ＿Ａの実体である、Ｒａｍｄｉｓｋ上のファイル／ｍｎｔ／ｒａｍ／ｆｉｌｅ＿Ａにデータがｗｒｉｔｅされる。
【００５３】
（実施例２）
実施例２は、ＡＰＩフッカにて、ＨＤＤ１５上のファイルをＲａｍｄｉｓｋ上に移動する際、Ｒａｍｄｉｓｋ上のファイルへのシンボリックリンクが作成されない点が、実施例１と異なり、他は全て実施例１に同じである。従って、実施例１と同一である場合は同一の符号を付し重複した説明は以下省略する。
【００５４】
以下、図５～７を用いて、実施例２に係る、ファイルディスクリプタを扱うシステムコールをフックするＡＰＩフッカの動作を説明する。
【００５５】
図５は、実施例２における、ＨＤＤ上のファイルの移動処理の手順を示すフローチャートである。この処理では、ＨＤＤ１５上のファイルに対して、１分間に１０００回以上のアクセスがあった場合、そのファイルをＲａｍｄｉｓｋに移動する。以降、そのファイルに対するアクセスがあった場合は、アクセス先を、移動したＲａｍｄｉｓｋ上に変更する。尚、この処理は、ＣＰＵ１６が、ＨＤＤ１５上にある本処理を実行するためのプログラムをＲＡＭ２４上で展開することにより実行される。
【００５６】
図５では、ＡＰＩフッカを用いて、ファイルディスクリプタを扱う既存のシステムコールの実行に先立ち行われる動作として、アクセス頻度の検出と、Ｒａｍｄｉｓｋへのファイル移動という動作が付加される点については、実施例１に同じである。実施例２ではこれに加え、Ｒａｍｄｉｓｋ上にファイルを移動させた後、ＨＤＤ１５上のファイルへのアクセス先を、Ｒａｍｄｉｓｋ上に変更する動作が付加される。これを行う一つの方法としては、例えば、Ｒａｍｄｉｓｋ上にファイルを移動した時点で、ｏｐｅｎ（）システムコールにより、Ｒａｍｄｉｓｋ上のファイルのファイルディスクリプタを取得しておくという方法がある。これにより、以降、ＨＤＤ１５上のファイルに対するファイルアクセスがあった場合は、上記Ｒａｍｄｉｓｋ上のファイルのファイルディスクリプタに差し替えてファイルアクセスを行うことができる。
【００５７】
本実施例によれば、ＨＤＤ１５上のファイルへのアクセス頻度が一定値を超過した時点で、アプリケーションに意識させることなく、そのファイルを、Ｒａｍｄｉｓｋに移動することができる。
【００５８】
図５において、まず、ＣＰＵ１６は、ステップＳ２００にて、ＡＰＩフッカにより、ファイルアクセス頻度が１分間に１０００回を超過するファイルの検出を行う。ここで、ファイルが検出されない場合、ステップＳ２３０の処理に移る。
【００５９】
ステップＳ２００にて、ファイルが検出された場合は、ステップＳ２１０に移り、ファイルがＨＤＤ１５上のものか否かを判定する。ここで、ＨＤＤ１５上のものでなければ、ステップＳ２３０の処理に移る。
【００６０】
ステップＳ２１０にて、ファイルがＨＤＤ１５上のものであれば、ステップＳ５２０に移り、ＡＰＩフッカにより、そのファイルをＲａｍｄｉｓｋに移動する。この時併せて、ＡＰＩフッカにより、以降、そのファイルに対するアクセスが発生した場合のアクセス先をＲａｍｄｉｓｋ上に変更する。すなわち、アクセス先を移動後のＲａｍｄｉｓｋ上のファイルに差し替えるようにしておく。その後、ステップＳ２３０の処理に移る。
【００６１】

ステップＳ２３０では、ＡＰＩフッカと同じＡＰＩ名である既存のシステムコールを実行し、本処理を終了する。

【００６２】

これらの処理により、ＨＤＤ１５上にあるファイルへのアクセス頻度が、１分間に１０００回を超過した場合、アプリケーションに意識させることなく、アクセス先をＲａｍｄｉｓｋ変更する。これにより、ＨＤＤ１５への大量のアクセスを抑止することができ、ＨＤＤ　Ｄｕｔｙ比率を低減することが可能となる。

【００６３】

図６は、実施例２における、ファイル再移動処理の手順を示すフローチャートである。この処理では、図５の処理によりＲａｍｄｉｓｋに移動したファイルへのアクセスが、１分間に１０００回のアクセスを下回った場合、そのファイルがＨＤＤ１５上に戻される。尚、この処理は、ＣＰＵ１６が、ＨＤＤ１５上にある本処理を実行するためのプログラムをＲＡＭ２４上で展開することにより実行される。

【００６４】

図６では、ＡＰＩフッカを用いて、ファイルディスクリプタを扱う既存のシステムコールの実行に先立ち行われる動作として、アクセス頻度の検出と、ＨＤＤ１５へのファイル移動という動作が付加される点については、図３に同じである。実施例２ではこれに加え、ＨＤＤ１５上にファイルを戻した後、ＨＤＤ１５上のファイルへのアクセス先を、Ｒａｍｄｉｓｋ上に変更させなくする動作が付加される。これを行う一つの方法としては、例えば、ＨＤＤ１５上にファイルを戻した時点で、ｃｌｏｓｅ（）システムコールにより、Ｒａｍｄｉｓｋ上のファイルのファイルディスクリプタをクローズするという方法がある。これにより、以降、ＨＤＤ１５からＲａｍｄｉｓｋ上に移動されたファイルへのアクセスであっても、本来の通りに、ＨＤＤ１５上のファイルのファイルディスクリプタを用いて、ファイルアクセスが行われる。

【００６５】

ＣＰＵ１６は、ステップＳ３００にて、ＡＰＩフッカにより、アクセス頻度が１分間に１０００回を超過していたものが超過しなくなったファイルの検出を行う。ここで、ファイルが検出されない場合、ステップＳ２３０の処理に移る。

【００６６】

ステップＳ３００にて、ファイルが検出された場合は、ステップＳ６１０に移り、ＡＰＩフッカにより、Ｒａｍｄｉｓｋ上のその検出されたファイルをＨＤＤ１５に戻す。この時併せてＡＰＩフッカにより、以降、そのファイルに対するアクセスがあった場合のアクセス先をＨＤＤ１５に戻す。引き続き、ステップＳ２３０の処理に移る。

【００６７】

ステップＳ２３０では、ＡＰＩフッカと同じＡＰＩ名である既存のシステムコールを実行し、本処理を終了する。

【００６８】

これらの処理により、図５の処理によりＲａｍｄｉｓｋ上に移動したファイルへのアクセス頻度が、１分間に１０００回を下回った場合、そのファイルをＨＤＤ１５に戻すことで、容量に制限の多いＲａｍｄｉｓｋを有効に利用することが可能となる。

【００６９】

図７は、実施例２における、シャットダウン開始時のファイル再移動処理の手順を示すフローチャートである。この処理では、図５の処理によりＲａｍｄｉｓｋ上に移動された全てのファイルは、シャットダウン開始時にＨＤＤ１５上に戻される。尚、この処理は、ＣＰＵ１６が、ＨＤＤ１５上にある本処理を実行するためのプログラムをＲＡＭ２４上で展開することにより実行される。

【００７０】

図７では、ＡＰＩフッカを用いて、ファイルディスクリプタを扱う既存のシステムコールの実行に先立ち行われる動作として、以下の４つの動作が付加される。具体的には、第１の動作として、シャットダウン開始の検出という動作が付加され、第２の動作として、

Ｒａｍｄｉｓｋに移動したファイルの検出という動作が付加され、第３の動作として、ＨＤＤ１５へのファイル移動という動作が付加される。さらに第４の動作として、ＨＤＤ１５上にファイルを戻した後、ＨＤＤ１５上のファイルへのアクセス先を、Ｒａｍｄｉｓｋ上のファイルに変更させなくするという動作が付加される。その他の、ＡＰＩフッカの特徴については、図５の場合と同様である。

【００７１】

ＣＰＵ１６は、ステップＳ４００にて、ＡＰＩフッカにより、シャットダウンが開始されたかを検出する。シャットダウンが開始されていない場合は、そのまま本処理を終了する。

【００７２】

ステップＳ４００で、シャットダウンの開始が検出された場合は、ステップＳ４１０に移り、図５の処理によりＲａｍｄｉｓｋに移動したファイルが存在するかを検出する。ここで、ファイルが検出されない場合は、そのまま本処理を終了する。

【００７３】

ステップＳ４１０で、Ｒａｍｄｉｓｋに移動したファイルが検出された場合は、ステップＳ６１０に移り、ＡＰＩフッカにより、Ｒａｍｄｉｓｋ上のその検出されたファイルをＨＤＤ１５に戻す。これと併せてＡＰＩフッカにより、以降、そのファイルに対するアクセスがあった場合のアクセス先をＨＤＤ１５に戻す。その後、ステップＳ４００の処理に移る。

【００７４】

これらの処理により、シャットダウン開始時に、図５の処理によりＲａｍｄｉｓｋ上に移動した全てファイルをＨＤＤ１５に戻すことで、Ｒａｍｄｉｓｋ上のファイルがシャットダウンによって揮発しないようにすることが可能となる。

【００７５】

（実施例３）

実施例３は、ＡＰＩフッカにて、１分間に１０００回以上のアクセスがあったＨＤＤ１５上のファイルのサイズが、Ｒａｍｄｉｓｋ容量の１０％以上であった場合の処理が実施例１と異なる。具体的には、かかる場合にそのファイルへのアクセスにより追記があったときは、その追記がされたデータのみをＲａｍｄｉｓｋに移動する。また、かかる場合であっても上記追記がなかった場合は、そのファイルを一定サイズのデータに分割し、その分割されたデータのうちファイルアクセスのあった箇所のデータのみをＲａｍｄｉｓｋに移動する。他は、実施例１に同じである。従って、実施例１と同一である場合は同一の符号を付し重複した説明は以下省略する。

【００７６】

以下、図８～１０を用いて、実施例３に係る、ファイルディスクリプタを扱うシステムコールをフックするＡＰＩフッカの動作を説明する。

【００７７】

図８は、実施例３における、ＨＤＤ上のファイルの移動処理の手順を示すフローチャートである。この処理では、ＨＤＤ１５上のファイルに対して、１分間に１０００回以上のアクセスがあった場合、そのファイルサイズが、Ｒａｍｄｉｓｋ容量の１０％以上であるかを判定する。この判定の結果、ファイルサイズが、Ｒａｍｄｉｓｋ容量の１０％以上である場合であって、そのファイルへのアクセスにより追記があったときは、その追記がされたデータのみをＲａｍｄｉｓｋに移動する。また、この判定の結果、ファイルサイズが、Ｒａｍｄｉｓｋ容量の１０％以上である場合であっても、かかる追記がなかった場合は、ファイルを一定サイズのデータに分割する。その後、その分割されたデータのうちファイルアクセスのあった箇所のデータのみをＲａｍｄｉｓｋに移動する。尚、その分割対象となるファイルのサイズが、Ｒａｍｄｉｓｋ容量の１０％未満であった場合は、実施例１と同じ処理が行われる。また、この処理は、ＣＰＵ１６が、ＨＤＤ１５上にある本処理を実行するためのプログラムをＲＡＭ２４上で展開することにより実行される。

【００７８】

図８では、ＡＰＩフッカを用いて、ファイルディスクリプタを扱う既存のシステムコールの実行に先立ち行われる動作として、実施例１と同様、以下の３つの動作が付加される。具体的には、アクセス頻度の検出と、シンボリックリンクの作成と、Ｒａｍｄｉｓｋへのファイル移動という動作である。実施例３ではこれに加え、以下の３つの動作が付加される。具体的には、第１の動作として、アクセス頻度が１分間に１０００回以上となったファイルサイズの取得という動作が付加される。次に、第２の動作として、前記サイズがＲａｍｄｉｓｋ容量の１０％を超過しているかの検出という動作が付加される。最後に、第３の動作として、ファイルを一定サイズのデータに分割した場合にファイルアクセスのあった箇所のデータ、もしくは、追記されたデータのみのＲａｍｄｉｓｋへの移動という動作が付加される。これを行う一つの方法としては、ファイルアクセスの際、ファイルへの追記があった場合は、まず、その追記されたデータを全てＲａｍｄｉｓｋ上に置く。以降、そのファイルへのアクセスは、ｌｓｅｅｋ（）により判明するファイルアクセス先のオフセットに応じて行う。これにより、ＨＤＤ１５上のファイル、もしくは、Ｒａｍｄｉｓｋ上に移動した上記追記されたデータにアクセス先を振り分けることができる。また、ファイルアクセスの際、ファイルへの追記がなかった場合は、４ＫＢｙｔｅのサイズのデータにファイルを分割し、その分割されたデータのうちファイルアクセスのあった箇所のデータのみ、Ｒａｍｄｉｓｋ上に置く。以降、そのファイルへのアクセスは、ｌｓｅｅｋ（）により判明するファイルアクセス先のオフセットに応じて行う。これにより、ＨＤＤ１５上のファイル、もしくは、上記分割後にＲａｍｄｉｓｋ上に移動したデータにアクセス先を振り分けることができる。尚、追記されたデータ、および、ファイルを４Ｋｂｙｔｅで分割した場合の、ファイルアクセスのあった箇所のデータのことを、以下総称して「部分ファイルデータ」という。尚、この際Ｒａｍｄｉｓｋに移したデータについて、ファイル全体か、部分ファイルデータかの区別を管理テーブルに記録しておく。さらに、Ｒａｍｄｉｓｋに移したデータが部分ファイルデータの場合は、対応するＲａｍｄｉｓｋ上のファイルディスクリプタ、及び、ＨＤＤ１５上のファイルとＲａｍｄｉｓｋ上の部分ファイルデータとのオフセットの対応を併せて管理テーブルに記録しておく。これにより、Ｒａｍｄｉｓｋ容量を圧迫し得るファイルに対しても、ファイルアクセスがあった部分ファイルデータのみがＲａｍｄｉｓｋに移動することとなる。このため、アプリケーションに意識させることなく、ＨＤＤ１５上のデータを、Ｒａｍｄｉｓｋに移動することができる。

【００７９】

図８において、まず、ＣＰＵ１６は、ステップＳ２００にて、ＡＰＩフッカにより、ファイルアクセス頻度が１分間に１０００回を超過するファイルの検出を行う。ここで、ファイルが検出されない場合、ステップＳ２３０の処理に移る。

【００８０】

ステップＳ２００にて、ファイルが検出された場合は、ステップＳ２１０に移り、ファイルがＨＤＤ１５上のものか否かを判定する。ここで、ＨＤＤ１５上のものでなければ、ステップＳ２３０の処理に移る。

【００８１】

ステップＳ２１０にて、ファイルがＨＤＤ１５上のものであれば、ステップＳ８１０に移り、ＡＰＩフッカによりそのファイルのファイルサイズを取得すると共に、取得したファイルサイズがＲａｍｄｉｓｋ容量の１０％未満か否かを検出する。ここで、１０％未満であれば、ステップＳ２２０に移り、ＨＤＤ１５にはそのファイルのシンボリックリンクのみ作成するとともに、そのファイル全体をＲａｍｄｉｓｋに移動する。

【００８２】

ステップＳ８１０にて、ファイルサイズがＲａｍｄｉｓｋ容量の１０％以上であった場合は、ステップＳ８２０に移り、ファイルに対するアクセスが、ファイルへの追記か否かを検出する。ファイルへの追記であった場合は、ステップＳ８３０に移り、ＡＰＩフッカにより、ファイルに追記されたデータのみＲａｍｄｉｓｋに置く。

【００８３】

ステップＳ８２０にて、ファイル追記ではなかった場合は、ステップＳ８４０に移り、ファイルを４Ｋｂｙｔｅのサイズのデータに分割し、ＡＰＩフッカにより、分割されたデータのうちファイルアクセスがあった箇所のデータのみＲａｍｄｉｓｋに置く。

【００８４】

ステップＳ２２０、ステップＳ８３０、ステップＳ８４０の処理後は、ステップＳ２３０にて、ＡＰＩフッカと同じＡＰＩ名である既存のシステムコールを実行し、本処理を終了する。

【００８５】

このように、ＨＤＤ１５上にあるファイルへのアクセス頻度が、１分間に１０００回を超過した場合、そのファイルのファイルサイズがＲａｍｄｉｓｋ容量の１０％以上と大きい場合がある。かかる場合であっても、これらの処理により、Ｒａｍｄｉｓｋを圧迫させることなく、ＨＤＤ１５への大量のアクセスを抑止することができる。これにより、ＨＤＤ　Ｄｕｔｙ比率を低減することが可能となる。

【００８６】

尚、本実施例では、ファイルのサイズがＲａｍｄｉｓｋ容量の１０％未満か否かを判定したが、１０％という数値に限定されるものではない。ファイルのサイズが、そのファイル全体をＲａｍｄｉｓｋに移動するとＲａｍｄｉｓｋを圧迫させる、Ｒａｍｄｉｓｋ容量の特定の割合未満であるかが判定されればよい。

【００８７】

図９は、実施例３における、ファイル再移動処理の手順を示すフローチャートである。この処理では、図８の処理により全体もしくはその一部が移動したファイルへのアクセスが、１分間に１０００回のアクセスを下回った場合、その移動したファイルの全体もしくは一部が、ＨＤＤ１５上に戻される。尚、この処理は、ＣＰＵ１６が、ＨＤＤ１５上にある本処理を実行するためのプログラムをＲＡＭ２４上で展開することにより実行される。

【００８８】

図９では、ＡＰＩフッカを用いて、ファイルディスクリプタを扱う既存のシステムコールの実行に先立ち行われる動作として、図３の処理と同じく、アクセス頻度の検出と、シンボリックリンクの削除と、ＨＤＤ１５へのファイル移動という動作が付加される。実施例３ではこれに加え、以下の２つの動作が付加される。具体的には、第１の動作として、Ｒａｍｄｉｓｋに移動したものが部分ファイルデータであった場合、ＨＤＤ１５上のファイルに対して、部分ファイルデータが存在する箇所については、部分ファイルデータを用いて更新する動作が付加される。また、第２の動作として、Ｒａｍｄｉｓｋ上の部分ファイルを削除する動作が付加される。これを行う一つの方法としては、例えば、アクセス頻度が１分間に１０００回を下回ったファイルの部分ファイルデータがＲａｍｄｉｓｋに存在していた場合は、その部分ファイルデータでＨＤＤ１５上のファイルを更新する方法がある。

【００８９】

図９において、まずＣＰＵ１６は、ステップＳ３００にて、ＡＰＩフッカにより、ファイルアクセス頻度が１分間に１０００回を超過していたものが超過しなくなったファイルの検出を行う。ここで、ファイルが検出されない場合、ステップＳ２３０の処理に移る。

【００９０】

ステップＳ３００にて、ファイルが検出された場合は、ステップＳ９１０の処理に移り、管理テーブルの参照により、図８の処理によりＲａｍｄｉｓｋに移動したのは部分ファイルデータか、ファイル全体かの検出を行う。ファイル全体であった場合は（ステップＳ９１０でＹＥＳ）、ステップＳ３１０に移り、ＡＰＩフッカにより、ＨＤＤ１５上のシンボリックリンクを削除すると共に、Ｒａｍｄｉｓｋ上のその検出されたファイル全体をＨＤＤ１５に戻す。

【００９１】

一方、Ｒａｍｄｉｓｋに移動したのは部分ファイルデータであった場合は（ステップＳ９１０でＮＯ）、ステップＳ９２０に移り、ＡＰＩフッカにより、Ｒａｍｄｉｓｋ上の部

分ファイルデータで、ＨＤＤ１５上のファイルデータを更新する。この時併せてＡＰＩフッカにより、Ｒａｍｄｉｓｋ上の部分ファイルデータを削除する。

【００９２】

ステップＳ３１０、ステップＳ９２０それぞれの処理後は、ステップＳ２３０にて、ＡＰＩフッカと同じＡＰＩ名である既存のシステムコールを実行し、本処理を終了する。

【００９３】

これらの処理により、図８の処理により、全体もしくはその部分ファイルデータがＲａｍｄｉｓｋ上に移動したファイルへのアクセス頻度が、１分間に１０００回を下回った場合、そのファイル全体もしくはその部分ファイルデータをＨＤＤ１５に戻す。これにより、容量に制限の多いＲａｍｄｉｓｋを有効に利用することが可能となる。

【００９４】

図１０は、実施例３における、シャットダウン開始時のファイル再移動処理の手順を示すフローチャートである。この処理では、図８の処理によりＲａｍｄｉｓｋ上に移動したファイルもしくは部分ファイルデータの全てが、シャットダウン開始時に、ＨＤＤ１５上に戻される。尚、この処理は、ＣＰＵ１６が、ＨＤＤ１５上にある本処理を実行するためのプログラムをＲＡＭ２４上で展開することにより実行される。

【００９５】

図１０では、ＡＰＩフッカを用いて、ファイルディスクリプタを扱う既存のシステムコールの実行に先立ち行われる動作として、以下の５つの動作が付加される。具体的には、第１の動作として、シャットダウン開始の検出という動作が付加され、第２の動作として、Ｒａｍｄｉｓｋに移動したファイルもしくは部分ファイルデータが存在するかの検出という動作が付加される。また、第３の動作として、シンボリックリンクの削除という動作が付加され、第４の動作として、ＨＤＤ１５へのファイル移動という動作が付加される。最後に、第５の動作として、部分ファイルデータを用いたファイルデータの更新という動作が付加される。その他の、ＡＰＩフッカの特徴については、図９の場合と同様である。

【００９６】

ＣＰＵ１６は、ステップＳ４００にて、ＡＰＩフッカにより、シャットダウンが開始されたかを検出する。シャットダウンが開始されていない場合は、そのまま本処理を終了する。

【００９７】

ステップＳ４００で、シャットダウンの開始が検出された場合は、ステップＳ１０１０に移り、ＡＰＩフッカにより、図８の処理でＲａｍｄｉｓｋに移動したファイル全体、もしくは、部分ファイルデータが存在するかを検出する。ここで、存在しない場合は、そのまま本処理を終了する。

【００９８】

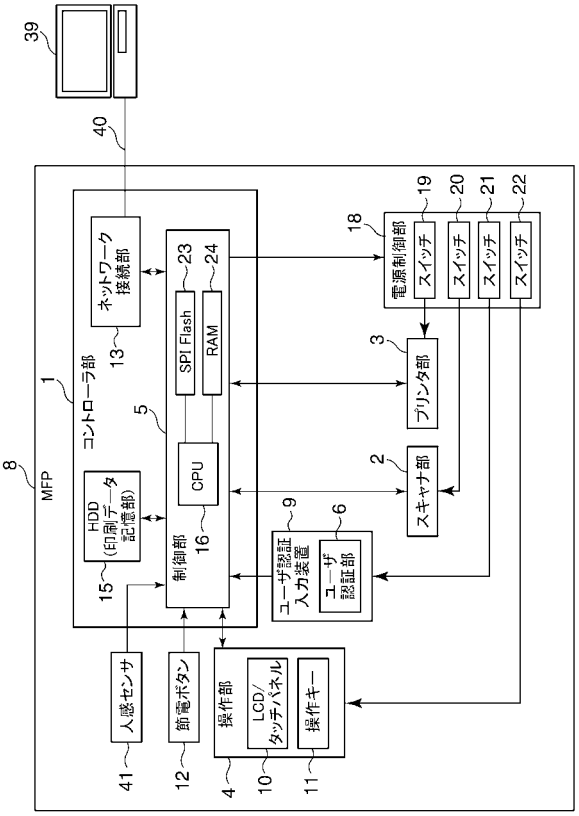
ステップＳ１０１０で、ファイル全体、もしくは、部分ファイルデータが検出された場合は、ステップＳ９１０に移り、部分ファイルデータ及びファイル全体のいずれが検出されたかを判定する。ここで、検出されたのはファイル全体であった場合は（ステップＳ１０１０でＹＥＳ）、ステップＳ３１０に移り、ＡＰＩフッカにより、ＨＤＤ１５上のシンボリックリンクを削除すると共に、Ｒａｍｄｉｓｋ上のその検出されたファイル全体をＨＤＤ１５に戻す。その後、ステップＳ４００の処理に戻る。一方、検出されたのは部分ファイルデータであった場合は（ステップＳ１０１０でＮＯ）、ステップＳ９２０に移り、ＡＰＩフッカにより、Ｒａｍｄｉｓｋ上のその検出された部分ファイルデータで、ＨＤＤ１５上のファイルデータを更新する。この時併せてＡＰＩフッカにより、Ｒａｍｄｉｓｋ上のその部分ファイルデータを削除する。その後、ステップＳ４００の処理に戻る。

【００９９】

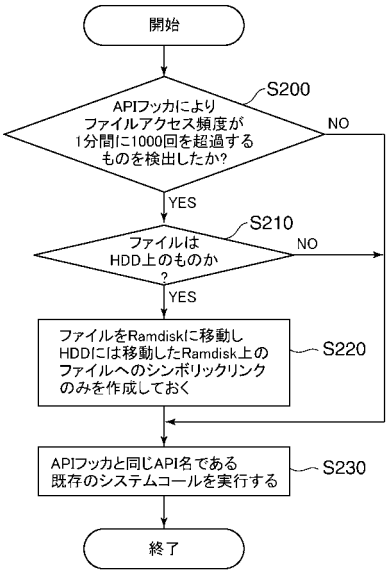
これらの処理により、シャットダウン開始時に、図８の処理でＲａｍｄｉｓｋ上に移動した全てのファイル若しくは部分ファイルデータをＨＤＤ１５に戻す。これにより、Ｒａｍｄｉｓｋ上の全てのファイル若しくは部分ファイルデータがシャットダウンによって揮発しないようにすることが可能となる。

【０１００】
尚、上記実施例１～３では、ＨＤＤ１５からＲＡＭ２４上のＲａｍｄｉｓｋにファイルや部分ファイルデータが移動するような構成であった。しかしながら、揮発性記憶媒体にファイルや部分ファイルデータが移動する構成であればＲａｍｄｉｓｋに限定されるわけではない。また、上記実施例１～３の処理はＭＦＰ８に対して行われたが、不揮発性記憶媒体及び揮発性記憶媒体を有し、ファイルディスクリプタを扱うシステムコールが実行される情報処理装置であればＭＦＰ８に限定されるわけではない。
【０１０１】
以上、本発明をその好適な実施形態に基づいて詳述してきたが、本発明はこれら特定の実施形態に限られるものではなく、この発明の要旨を逸脱しない範囲の様々な形態も本発明に含まれる。上述の実施形態の一部を適宜組み合わせてもよい。また、上述の実施形態の機能を実現するソフトウェアのプログラムを、記録媒体から直接、或いは有線／無線通信を用いてプログラムを実行可能なコンピュータを有するシステム又は装置に供給し、そのプログラムを実行する場合も本発明に含む。従って、本発明の機能処理をコンピュータで実現するために、該コンピュータに供給、インストールされるプログラムコード自体も本発明を実現するものである。つまり、本発明の機能処理を実現するためのコンピュータプログラム自体も本発明に含まれる。その場合、プログラムの機能を有していれば、オブジェクトコード、インタプリタにより実行されるプログラム、ＯＳステップＳに供給するスクリプトデータ等、プログラムの形態を問わない。プログラムを供給するための記録媒体としては、例えば、ハードディスク、磁気テープ等の磁気記録媒体、光／光磁気記憶媒体、不揮発性の半導体メモリでもよい。また、プログラムの供給方法としては、コンピュータネットワーク上のサーバに本発明を形成するコンピュータプログラムを記憶し、接続のあったクライアントコンピュータがコンピュータプログラムをダウンロードしてプログラムするような方法も考えられる。
【符号の説明】
【０１０２】
１　コントローラ部
４　操作部
５　制御部
８　ＭＦＰ
１２　節電ボタン
１３　ネットワーク接続部
１５　ＨＤＤ
１６　ＣＰＵ
２３　ＳＰＩ　Ｆｌａｓｈ
２４　ＲＡＭ

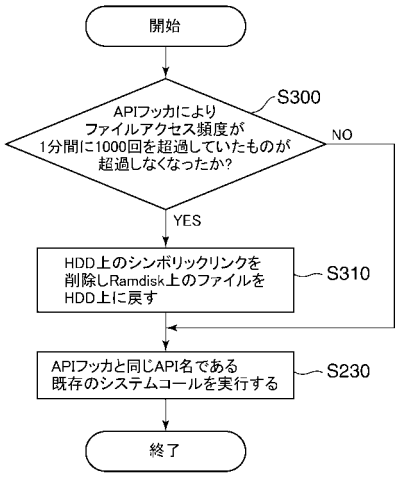
【図１】


39
40
MFP
8
コントローラ部
1
ネットワーク接続部
13
SPI Flash
23
RAM
24
CPU
16
制御部
5
HDD（印刷データ記憶部）
15
電源制御部
18
スイッチ
19
スイッチ
20
スイッチ
21
スイッチ
22
プリンタ部
3
スキャナ部
2
ユーザ認証入力装置
9
ユーザ認証部
6
人感センサ
41
節電ボタン
12
操作部
4
LCD/タッチパネル
10
操作キー
11

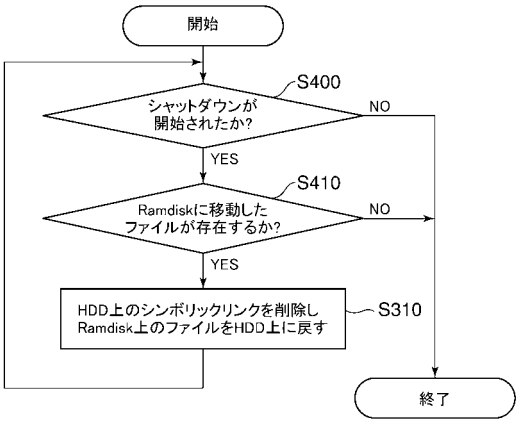
【図２】


開始
APIフッカにより
ファイルアクセス頻度が
1分間に1000回を超過する
ものを検出したか?
S200
NO
YES
ファイルは
HDD上のものか
?
S210
NO
YES
ファイルをRamdiskに移動し
HDDには移動したRamdisk上の
ファイルへのシンボリックリンク
のみを作成しておく
S220
APIフッカと同じAPI名である
既存のシステムコールを実行する
S230
終了

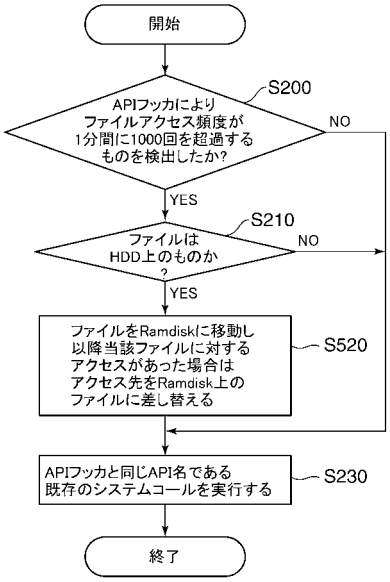
【図３】


開始
APIフッカにより
ファイルアクセス頻度が
1分間に1000回を超過していたものが
超過しなくなったか?
S300
NO
YES
HDD上のシンボリックリンクを
削除しRamdisk上のファイルを
HDD上に戻す
S310
APIフッカと同じAPI名である
既存のシステムコールを実行する
S230
終了

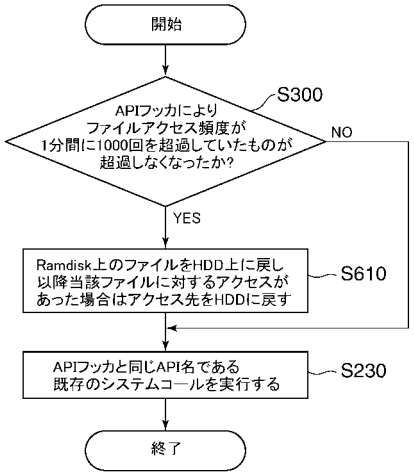
【図４】


開始
シャットダウンが
開始されたか?
S400
NO
YES
Ramdiskに移動した
ファイルが存在するか?
S410
NO
YES
HDD上のシンボリックリンクを削除し
Ramdisk上のファイルをHDD上に戻す
S310
終了

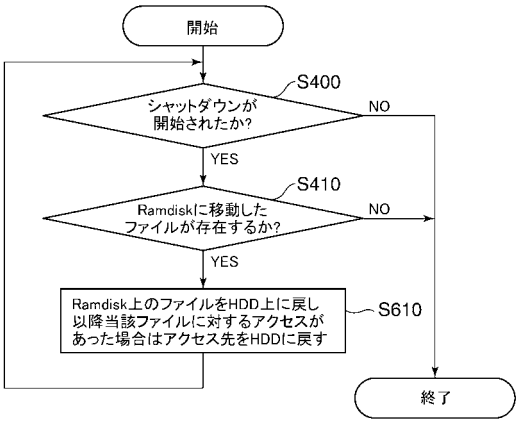
【図５】


開始
S200
APIフッカにより
ファイルアクセス頻度が
1分間に1000回を超過する
ものを検出したか?
NO
YES
S210
ファイルは
HDD上のものか
?
NO
YES
ファイルをRamdiskに移動し
以降当該ファイルに対する
アクセスがあった場合は
アクセス先をRamdisk上の
ファイルに差し替える
S520
APIフッカと同じAPI名である
既存のシステムコールを実行する
S230
終了

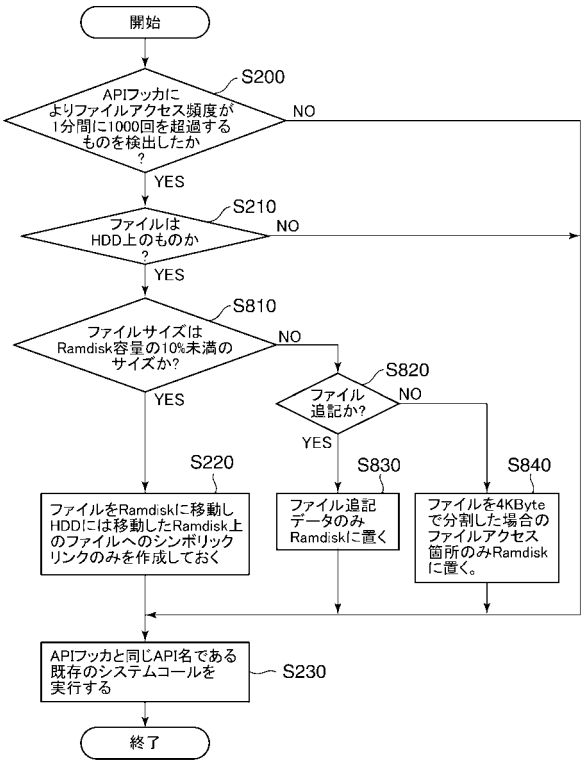
【図６】


開始
S300
APIフッカにより
ファイルアクセス頻度が
1分間に1000回を超過していたものが
超過しなくなったか?
NO
YES
Ramdisk上のファイルをHDD上に戻し
以降当該ファイルに対するアクセスが
あった場合はアクセス先をHDDに戻す
S610
APIフッカと同じAPI名である
既存のシステムコールを実行する
S230
終了

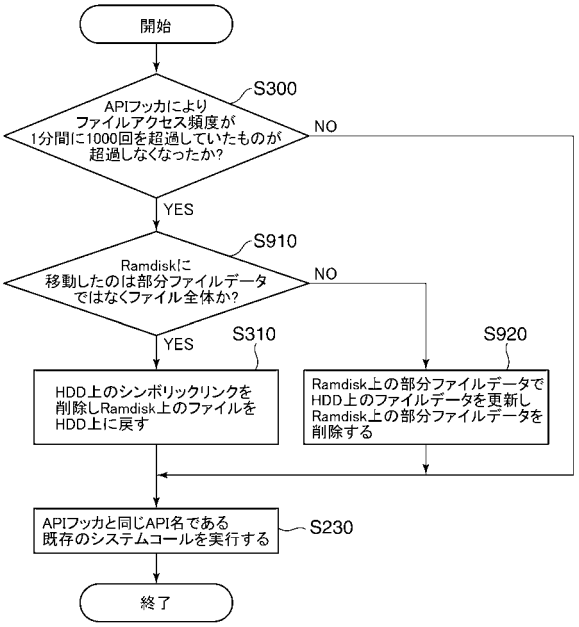
【図７】


開始
S400
シャットダウンが
開始されたか?
NO
YES
S410
Ramdiskに移動した
ファイルが存在するか?
NO
YES
Ramdisk上のファイルをHDD上に戻し
以降当該ファイルに対するアクセスが
あった場合はアクセス先をHDDに戻す
S610
終了

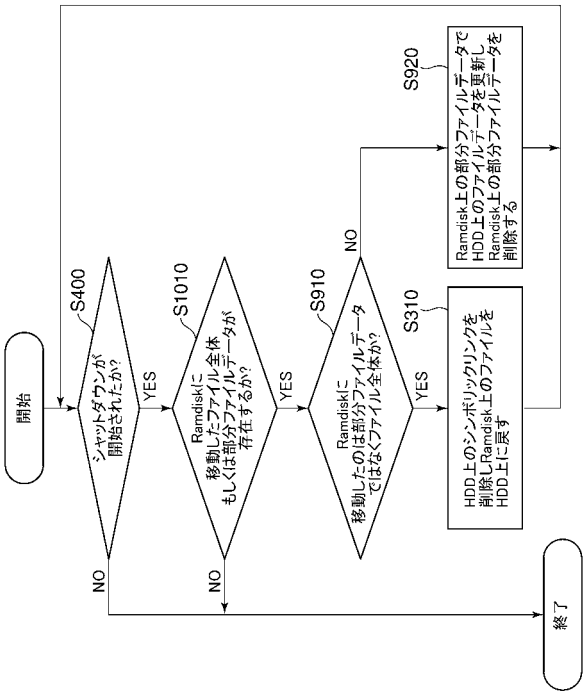
【図８】


開始
S200
APIフッカに
よりファイルアクセス頻度が
1分間に1000回を超過する
ものを検出したか
?
NO
YES
S210
ファイルは
HDD上のものか
?
NO
YES
S810
ファイルサイズは
Ramdisk容量の10%未満の
サイズか?
NO
YES
S820
ファイル
追記か?
NO
YES
S220
ファイルをRamdiskに移動し
HDDには移動したRamdisk上
のファイルへのシンボリック
リンクのみを作成しておく
S830
ファイル追記
データのみ
Ramdiskに置く
S840
ファイルを4KByte
で分割した場合の
ファイルアクセス
箇所のみRamdisk
に置く。
APIフッカと同じAPI名である
既存のシステムコールを
実行する
S230
終了

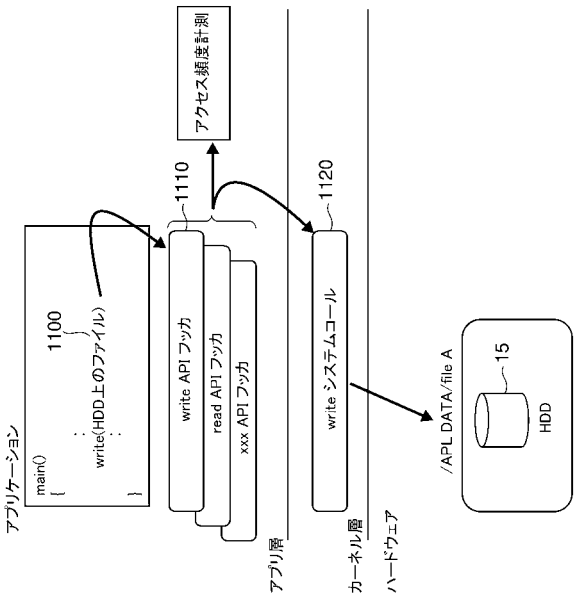
【図９】



【図１０】



【図１１】



【図１２】