ファイル毎のフックを設定可能なシステムの実装

1211001　青山 航　　岩崎研究室

1　背景

ユーザはフックを用いることで，独自の処理 (以下フック処理とする) を元のプログラム中に割り込ませることができる．ここでフックとは，プログラム中の特定の箇所に独自の処理を追加する機構のことである．システムコールに対してフックを行うことで，ユーザはカーネルのソースコードを変更せずにカーネル自体の機能を拡張 · 変更することができる．

ユーザがファイル毎に別々のフック処理を適用できれば有用である．例えば，重要なファイルにのみ暗号化して保存をする，特定のファイルにのみバックアップを作成する，怪しいファイルにのみウィルス検査をする，など様々な用途が考えられる．しかし，システムコールのフック処理はカーネル空間で実行されるため，フック処理に誤りがあるとカーネルがクラッシュするなど，カーネルが正常に動作しなくなる．例えば，以下のようなバグのあるプログラムをフック処理として実行してしまうとカーネルが正常に動作しなくなる．

- 配列をはみ出しての書き込みをする．
- 終了しないプログラムを実行する．

そのため，ユーザが定義した任意の処理をカーネル空間で実行させるのは，好ましくない．

2　目的と方針

本研究では，ファイル毎に別々のフックを安全にかけることが可能なシステムを実現することを目的とする．そのための方針として，Loadable Kernel Module (以下 LKM) を用いることで，システムコールのフック処理を実現する．LKM とは起動中のカーネルに対して，新しい機能を追加することを可能にする機構であり，追加された処理はカーネル空間で動作する．本研究では，カーネル空間で定義されているシステムコールの処理関数へのポインタを LKM を用いて書き換えることでシステムコールのフックを実現する．

さらに，ユーザが定義するフック処理はユーザ空間で実行させることで，カーネルクラッシュの危険のない安全な処理を提供する．ユーザ空間で前述のバグがあるプログラムを実行させても，エラーとして実行が中断されたり，プロセスに割り込んで終了しないプログラムを終了させることができる．

3　実行例

本システムを使用するために，ユーザは次の動作を行う必要がある．

- フック処理を実装する関数は予め作成しておく．
- 本研究で追加したシステムコールを用いてフック処理とファイルの対応関係を登録する．

専用のシステムコールが実行されると，カーネル空間内に対応関係が登録されたテーブル (フック関数対応表) が生成され，それ以降のファイルアクセス時にフック処理が自動的に適用される．図 1 は，ファイル「foo」に書き込みをするプロセス A とファイル「bar」に対して書き込みを行うプロセス B が，以下の対応をカーネルに登録した場合における本システムの実行例である．

- ファイル「foo」は `write` 処理の前に暗号化する．
- ファイル「foo」は `read` 処理の後に復号する．
- ファイル「bar」は `write` 処理の前にウィルス検査をする．

プロセス A は `write` システムコールを実行すると，カーネル内が本システムの LKM に処理が遷移する．LKM がフック関数対応表とシステムコールの種類，アクセスするファイル名からフックを行う必要があると判断し，ユーザ空間の暗号化処理にデータを渡す．(フックを後に処理する場合は先に本来の処理を行ってからフック処理をする) ユーザ空間にある暗号化処理は，カーネルからデータを渡されるとユーザが独自に定義した処理を行い，その結果を LKM に返す．LKM がフックする必要がないと判断するとデータを元のシステムコール処理に渡し，ファイルに操作を行う．同様にプロセス B では `write` システムコールの処理の前にウィルス検査を行い，その後に本来のシステムコールの処理をする．

このように，ユーザが定義した任意の処理が LKM によってファイル毎に異なる処理が実行されるようになる．

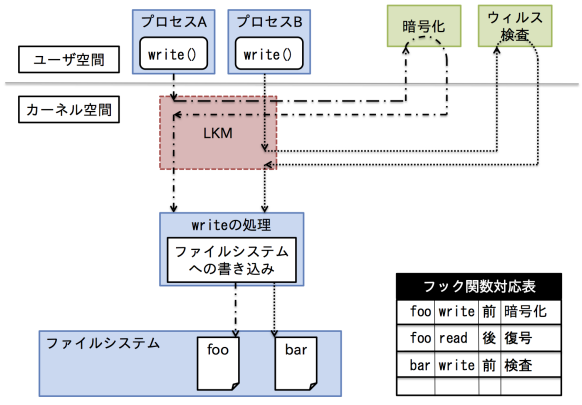

図 1　本研究で追加するシステムの動作例

4　関連研究

Szeredi らの設計した Filesystem in Userspace [1] は，これを利用することにより，ユーザ空間で動作するファイルシステムを実装できるシステムである．これを利用することにより，ファイルシステムにおけるファイル操作の処理を変更することができる．ユーザ空間でファイル操作処理を実行する点では本研究と同じであるが，本研究ではファイルシステムに対して処理を変更するのではなく，個々のファイルに対してユーザが定義したフック処理を実行することができるようなシステムを提供する．

5　現状と今後

これまで，Linux のカーネル構造に関する調査を行い，LKM を用いたシステムコールをフックする方法の確認した．今後は本研究におけるモジュールを実装してその有用性を立証する．さらに本システムを導入したことによるオーバーヘッドの評価を行う予定である．

参考文献

[1]「Filesystem in Userspace」 http://fuse.sourceforge.net