情報処理システム Ⅱ ・第 14 回

2024 年 7 月 19 日

今回の内容

14.1 計算機セキュリティ 14–1
14.2 OS における資源の保護 14–2
14.3 Unix 系 OS におけるユーザー管理とファイル保護 14–3

14.1 計算機セキュリティ

一般に、計算機システムには、次のような性質が期待されます。

完全性 情報の記憶や処理を正しく行えることです。ファイルに記憶したはずのデータがなくなったり、いつのまにか改変されていたりしてはいけませんし、計算結果が間違っていたりしては困ります。

機密性 情報が伝わる範囲を正しく制限できることです。特定のユーザだけしか知ってはいけない情報がその他のユーザ (全く関係のない部外者を含む) に漏洩しては困ります。

可用性 いつでも十分な性能を発揮できることです。全く使用できなかったり、十分な速度で動作しない状態になってしまうのは困りますし、たとえ、そのような状態になったとしても、できるだけ早く正常な状態に復帰できて欲しいものです。

計算機システムのこのような 3 つの性質を維持することを**計算機セキュリティ**と呼びます。

メモ

完全性、機密性、可用性を阻害する要因　これら 3 つの性質を阻害する要因には、次に挙げるように様々ものがあります。

1. ハードウェアの故障や災害等による物理的な破壊、停電
2. ユーザの操作ミス
3. ソフトウェアの不具合
4. 悪意あるソフトウェアの侵入
5. ネットワークを介した外部からの攻撃

これらの要因に対する対策も様々です。地震に備えて耐震性のある建物に計算機を設置したり、停電に備えて無停電電源装置 (UPS[1]) を使用したりするような、OS の枠を超えた方法をあります

[1] Uniterruptable Power Supply

が、定期的なデータのバックアップを取るための仕組みや、1 つのソフトウェアの不具合や、1 つの悪意のあるソフトウェアの侵入の影響が他の部分へ波及し難くなるような仕組みなど、OS が提供する機能の一部となっているものもあります。

メモ

14.2 OS における資源の保護

オペレーティングシステム (OS) の主要な役割は、計算機システムの資源を管理することですが、単に、複数のユーザプロセスが各種の資源を利用する際の交通整理をするだけでなく、計算機セキュリティーを目的として、各種の資源を**保護**するための仕組みを提供するのが一般的です。計算機ネットワークのことを考えると、OS が稼働している 1 つの計算機システム上の資源だけでなく、もっと広い範囲でその保護を扱うことになりますが、ここでは、1 つの OS が管理している単一の計算機システム上の資源の保護に限定して説明します。

主体、対象、操作　OS による各種の資源の保護は、保護する「**対象**」に対する何らかの「**操作**」と、その対象に対してその操作を行なう「**主体**」との間の関係の管理であると捉えることができます。例えば、あるファイルを「対象」に、そのファイルに対する書き込みという「操作」を行うユーザプロセスが「主体」であり、OS (カーネル) は、これを許すのか許さないのかを管理することで資源の保護を行ないます。

メモ

保護の設定方法　OS は、この管理を何らかの設定に基づき行うことになりますが、1 つの計算機システムで OS が管理している「対象」と「操作」は無数にあり、それを行う可能性のある「主体」も無数にありますので、それらすべての組み合わせに対して、許可するのか許可しないのかを設定しておくのは事実上不可能です。そこで、通常は、次の 2 つのいずれかか、この 2 つを組み合わせた方法が用いられます。

アクセス制御リスト　各「対象」に対して、どの「主体」のどの「操作」を許すのかを設定する。

資格リスト　各「主体」に対して、どの「対象」に対するどの「操作」を許すのか設定する。

多くの OS で、「対象」に対して実際に「操作」を行うのは、各ユーザプロセスです。しかし、ユーザプロセスは誕生と消滅を繰り返しますので、アクセス制御リストにおいて、特定のプロセスを指定して、許される「操作」を (各「対象」に) 設定することは困難となります。このため、通常は、**ユーザ**(やそのグループ) という概念を導入し、このユーザを「主体」と捉え、プロセスはその代理人として「操作」を行っていると考えて、「対象」の保護を行います。

一方、資格リストを用いる場合は、1 つの資格リストを主体となるプロセスに与えて置けば、それをプロセス間で引き継いで行くこともできますので、必ずしもユーザという概念は必要ではありませんが、通常は、やはりユーザという概念が導入されていて、まず、ユーザに対する資格リストが作成されて、それを持ったプロセスが生成されるのが一般的です。そのような場合でも、プロセスが独自に、途中で資格リストに新たな資格を追加したり、必要のない資格を意図的に捨ててしまうようなことがあります。

メモ

14.3 Unix 系 OS におけるユーザー管理とファイル保護

OS による資源の保護の一例として、Unix 系の OS におけるファイルの保護の基本的な枠組みについて紹介します[2]。

ユーザとグループ Unix 系 OS で資源の保護を行う際の「主体」となるのは、**ユーザ**とその**グループ**という概念です。カーネル内では、それぞれ、ユーザ ID と グループ ID と呼ばれる非負の整数で識別され、各ユーザプロセスには、1 つのユーザ ID といくつか (1 つ以上) のグループ ID のリストが、プロセス属性の 1 つとして保持されます。ユーザやグループは、それぞれ、その計算機システムを利用する人間やそのグループに対応しますが、これらの他に、OS 自身がサーバプログラムなどの実行のためにあらかじめ用意しているものあります。ユーザについては、ユーザごとに

ユーザ名、パスワード、ユーザ ID、グループ ID、ホームディレクトリ、ログインシェル[3]

といった情報が、グループについては、グループごとに

グループ名、パスワード、グループ ID、そのグループに属するユーザ

といった情報が OS に登録されています[4]。

---

[2] ここで紹介するのは、初期の Unix から採用されている仕組みで、現在の Unix 系 OS では、ファイルシステムごとに様々な拡張が行われています。

[3] このユーザーのために起動されるシェル

[4] パスワードに関する情報は存在しない場合もあり、そのような場合、後述のログイン処理では、別の方法でユーザの認証を行います。パスワードが含まれる場合には、暗号化されています。

ログイン処理　Unix 系 OS のカーネルは、立ち上げの処理が完了すると、**ログイン**処理を行うためのサーバプログラムをユーザープロセスの 1 つとして起動します。このプロセスのユーザ ID は 0 です。Unix 系 OS は、ユーザ ID の 0 を、すべての権限を持つ**スーパユーザ**[5]として扱います。ユーザ ID が 0 のプロセスは万能で[6]、自分自身のプロセスのユーザ ID やグループ ID リストを変更することもできます。

ログイン処理を行うサーバプログラムは、これからその計算機システムを利用しようとしている利用者にユーザ名とパスワードを入力してもらい、これを OS に登録されている情報と照合します。一般に、このような、利用者が本人であることを確認する手続きを**ユーザ認証**と呼びます。

ユーザ認証に成功した (ログイン処理を行う) サーバプロセスは、`fork` した後、自分自身のプロセスのユーザ ID やグループ ID リストを、その利用者の (登録されている) ユーザ ID とグループ ID に変更し、ログイン後の GUI (メニューなど) を表示するプログラムを起動します。このプログラムから、例えば、端末エミュレータやシェルなどが起動されることになりますが、これらのプロセスのユーザ ID やグループ ID リストは、すべてログインした利用者のものとなります[7]。ユーザ ID 0 (スーパユーザ) のプロセスは万能で、自分自身のユーザ ID とグループ ID を好きなものに変更することができますが、一旦、ユーザ ID が 0 でなくなってしまえば、その新しいユーザ ID (一般ユーザ) としての権限しか持っていませんので、ユーザ ID を 0 に戻すことはできません。

メモ

ファイル保護　Unix 系 OS でのファイルの保護は、非常に単純な形のアクセス制御リスト[8]を、それぞれにファイルに対して設定することで行います。このアクセス制御リストは、次のような表の空欄を、それぞれ「許可」あるいは「拒否」で埋めたものに対応します。

操作	ユーザ	グループ	その他
読み出し			
書き込み			
実行			

[5]そのユーザー名は、通常 `root` です。

[6]と言っても、あくまでユーザープロセスであり、CPU の特権モードで実行されているわけではありません。

[7]`fork` や `execve` を行っても、プロセスのユーザ ID やグループ ID は変りません。

[8]Unix 系 OS のファイルシステムによっては、任意のユーザやグループに対して許可や拒否を設定できるような、もっと複雑なアクセス制御リストが利用できる場合もあります。そのような場合、「アクセス制御リスト」と言えば、そちらの設定を指し、ここで紹介するアクセス許可モード (保護モード) はアクセス制御リストとは呼びませんので注意してください。

この表の設定はファイルの**アクセス許可モード**、あるいは**保護モード**と呼ばれ、各ファイルのメタデータとして記録されています。また、各ファイルには、これとは別に、そのファイルの所有者となるユーザ ID とグループ ID が、やはりメタデータとして記録されています。

ファイルアクセスの許可/拒否の判定　ユーザプロセスがファイルを open しようとする際には、次の 4 つの情報に基づいて、その open を許可するか拒否するかをカネールが決定します、

1. open 関数の第 2 引数 (読み出しのみ/書き込みのみ/読み書き両方の別)
2. そのプロセスのユーザ ID とグループ ID リスト
3. ファイルのユーザ ID とグループ ID
4. ファイルのアクセス許可モード (保護モード)

許可するか拒否するかは、読み出しと書き込みのそれぞれの操作について判定され、許可されている操作のみが (open の第 2 引数で) 要求されている場合にのみ、open が成功します[9]。読み出しや書き込みの判定は、それぞれ、アクセス許可モードの「読み出し」と「書き込み」の行を使って判定されますが、

- プロセスとファイルのユーザ ID が一致している場合は「ユーザ」の列の設定が、
- そうでなくて、ファイルのグループ ID がプロセスのグループ ID のいずれかと一致している場合は「グループ」の列の設定が、
- これら 2 つのいずれでもない場合は「その他」の列の設定が

使用されます。

また、そのファイルを execve システムコールなどで起動しようとする際には、アクセス許可モードの表の「実行」の行を使って同様に判定されます。対象がディレクトリの場合は、「読み込み」は、そのディレクトリに置かれているファイルなどの一覧の読み出しを、「書き込み」は、そのディレクトリへのファイルなどの追加や削除を、「実行」は、そのディレクトリをパス名の一部に使用できるかどうかの判定に使用されます。

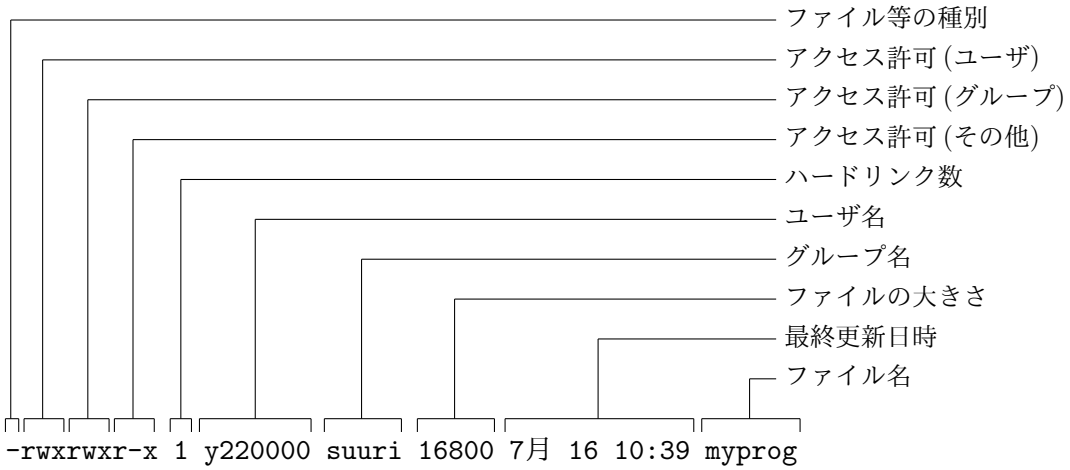
メモ

[9] open の第 2 引数で要求した「読み出し」や「書き込み」が一旦許可されれば、たとえ、open 中に、そのファイルのアクセス許可モードが変更されても、そのファイルが close されるまでは、引き続き「読み出し」や「書き込み」は許可されたままとなります。

ls コマンドによるファイルのアクセス許可リストの表示　次の例のように、ls コマンドを -l オプションを付けて実行すると、ファイルのユーザ名やグループ名とともに、アクセス許可モードの設定を表示することができます。

```
y220000@s01612h001:~$ ls -l
合計 101
-rwxrwxr-x 1 y220000 suuri 16800  7月 16 10:39 myprog
-rw-rw-r-- 1 y220000 suuri   141  7月 15 22:32 myprog.c
drwxrwxr-x 2 y220000 suuri    50  6月 21 15:07 testdir
y220000@s01612h001:~$
```

1 行ずつ各ファイル (やディレクトリ) のメタデータに関する情報が表示されていますが、それぞれの欄の意味は次の通りです。アクセス許可モード中の r、w、x は、それぞれ、「読み出し」、「書き込み」、「実行」という操作の「許可」を表しており、対応する箇所の - は「拒否」を表しています。



情報処理システム Ⅱ ・第 14 回・終わり