

Fundamentos de Sistemas de Operação

Unix Windows NT Netware Macos DOS/V/S Vax/VMS
Linux Solaris HP/UX AIX Mach Chorus

*Controle de Acessos no UNIX:
Conceitos, Mecanismos (e alguns comandos)*

AAA: Autenticação, Autorização, Acesso (1)

□ Autenticação

- *Processo pelo qual uma entidade (utilizador, programa em execução) quando desafiado, fornece informação que permite ao sistema concluir que é, de facto, quem diz ser.*
- *Exemplos:*
 - Um utilizador, no login, fornece as suas credenciais: “nome de utilizador”, ou “nome da conta” e senha (password). O sistema, ao verificar que reconhece o par fornecido, considera o processo de autenticação concluído com sucesso.
 - Um programa em execução (processo), tenta abrir um ficheiro, e o sistema “pede ao processo” as suas credenciais.
- *Nota: 2FA, autenticação usando 2 factores implica, p. ex., usar uma senha e um código fornecido por telefone...*

AAA: Autenticação, Autorização, Acesso (2)

□ Autorização

- *Processo pelo qual uma entidade (programa em execução) já autenticada que pretende realizar uma operação sobre um dado recurso (ficheiro, zona de memória, ...) quando desafiada, fornece informação que permite ao sistema decidir, usando a informação que tem sobre o recurso, se autoriza ou não a realização da operação.*
- *Exemplos:*
 - Um processo tenta um `open()` de um ficheiro.
 - Uma thread tenta um aceder a uma zona de memória no EE do processo.

AAA: Autenticação, Autorização, Acesso (3)

□ (Controle de) Acesso

- *Mecanismo pelo qual o sistema: a) define as restrições de acesso ao recurso, e b) garante que elas são respeitadas.*
- *Exemplos:*
 - Um dado ficheiro X só pode ser aberto para leitura.
 - Um dado processo Y só pode ser morto por um seu ancestral (pai, avô,...).

Unix: controle de acessos (1)

□ *Modelo básico: permissões*

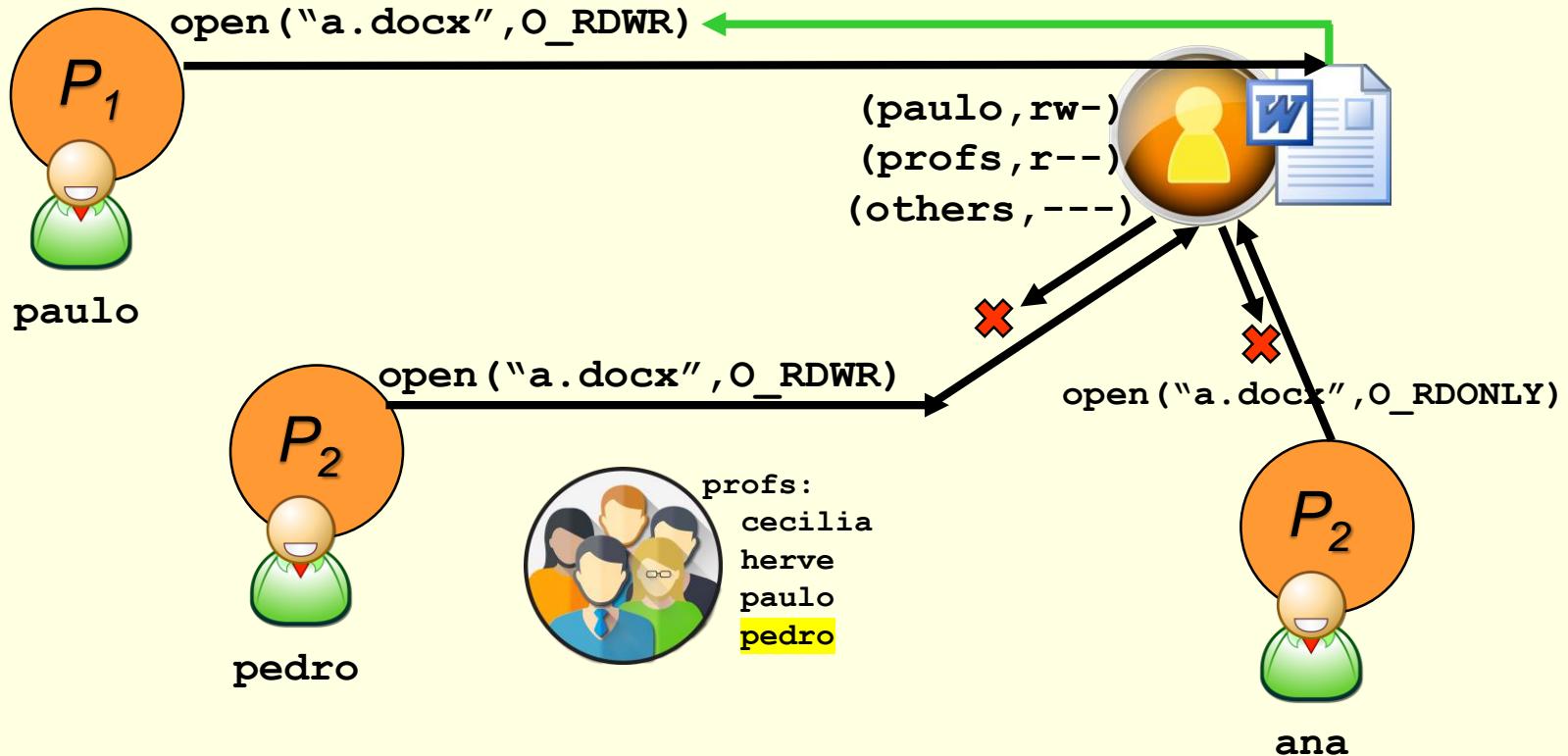
- O acesso a um recurso é mediado pelos (valores dos) seguintes atributos:
 - Utilizador **dono** (owner) **do recurso**
 - Grupo (de utilizadores)
 - Permissões: (R) **ead**, (W) **rite,e** (X) **ecute**

□ *Aplicação de uma “chave de segurança” a um recurso:*

- “Chave” formada por 3 pares: (Dono, Permissões), (Grupo, Permissões), (Outros, Permissões)
 - Exemplo: (paulo,rwx), (prof, rw-), (others,---)

Unix: controle de acessos (2)

Modelo operacional



Unix: controle de acessos (3)

- Ficheiros de suporte (a.k.a. “bases de dados”) [simplificados]
 - Utilizadores (ou “contas”): /etc/passwd
 - Cada linha começa com o “nome do utilizador”, ou login; segue-se o UID (User ID), nº inteiro único que identifica o utilizador, e o GID (Group ID), nº inteiro que identifica o grupo principal ao qual o utilizador pertence, etc...
 - Grupos: /etc/groups
 - Cada linha começa com o “nome do grupo”, ou login; segue-se o GID (Group ID), nº inteiro que identifica o grupo e finalmente uma lista de “nomes” que identificam os utilizadores incluídos nesse grupo.

Nota: Não esquecer – em cada inode estão registados o UID do dono, o GID do grupo que tem acesso ao ficheiro, e os 9 bits correspondentes aos 3 conjuntos de permissões: se 1 a permissão foi dada, se 0 foi negada.

Unix: controle de acessos (4)

□ Permissões: interpretação de casos particulares

- Ficheiros: permissão **x**
 - Se o ficheiro é um programa executável (“um binário”), então pode ser executado. Se o ficheiro é um “script” para ser executado por um interpretador (esse sim, “um binário”), então pode ser executado invocando apenas o nome do ficheiro; senão, é preciso chamar o interpretador e passar-lhe o nome do script como argumento.
 - Para outros “tipos” de ficheiros, a permissão **x** é ignorada.
- Directorias:
 - **Permissão x**: Permite “atravessar” a directória (o nome pode estar num path), ou entrar nela (com `cd`). **Permissão r**: Permite listar os nomes dos ficheiros existentes na directória. **Permissão w**: Permite criar, mudar o nome ou apagar ficheiros existentes na directória.

Unix: controle de acessos (5)

□ Permissões e a criação de ficheiros

- Criação por omissão: `creat("ficheiro", 0666)`
 - Num primeiro olhar, pareceria que o resultado seria a criação de um ficheiro de nome “ficheiro”, com as permissões `rw` para o dono, grupo e outros. Contudo uma verificação (p.ex., com o comando `ls -l`) mostra que as permissões ficaram com 0664 (`rw-rw-r--`). Porquê?
- Um dos atributos de um processo é o `umask`, que é um valor (bitmask) a aplicar às permissões especificadas num `creat` aquando da criação de um ficheiro de forma a que as indicadas na mask sejam as que ser quer negar. O valor por omissão da `umask` é 0002, o que faz com que as permissões resultantes sejam
 $mode = 0666 \& \sim 0002$, ou seja $mode = 0664$

Unix: controle de acessos (6)

□ Comandos importantes [simplificados]

- Mudar as permissões: **chmod <token> ficheiro**
 - Formato “amigável”: **chmod ugo+r ficheiro** - acrescentar permissão de *read* para “*todos*”: *utilizador (dono)*, *membros do grupo*, e *todos os outros*.
 - Formato “numérico”: **chmod 760 ficheiro** – atribuir *todas* as permissões ao *dono* (7=111), *read* e *write* aos *membros do grupo*, e *nenhuma* a *todos os outros*.
- Mudar o dono: **chown <nome> ficheiro**
 - Exemplo: **chown pedro ficheiro** – o dono do ficheiro passou a ser o *utilizador “pedro”*. Nota: cuidado – o antigo dono pode perder o acesso...

Unix: controle de acessos (7)

□ *Omnipotente ROOT*

- As operações efectuadas sobre ficheiros por processos lançados pelo utilizador `root` (`UID=0`) não são sujeitos ao mecanismo de autorização, são automaticamente permitidas.
- As operações realizadas sobre processos (`nice` ou `renice` para aumentar a prioridade, `kill`, etc.) são também permitidas.
- Contudo, há situações inultrapassáveis, mesmo para `root` – p.ex., um processo root não pode aceder a uma página inválida do seu EE, ou executar instruções privilegiadas (como referimos anteriormente ☺)

Demo: controle de acessos

- Ver os ficheiros `/etc/passwd` e `/etc/groups`
- Listar e alterar permissões e comprovar os efeitos das alterações
- `umask` e criação de ficheiros

Unix: ACL

□ *O mecanismo das ACLs (Access Control Lists)*

- *O mecanismo anteriormente exposto, “Unix permissions” é insuficiente para as necessidades de controle de acessos que se foram manifestando ao longo dos tempos...*
- *Uma das tentativas de melhorar a situação foi a introdução de listas de controle de acessos, que permitem um controle muito mais fino de quem tem acesso, e que operações pode executar...*
- *Contudo, o mecanismo é um pouco mais complexo e não o estudaremos aqui... ☺*