Namespaces do Linux

Conceito e gerenciamento de namespaces no kernel Linux

Integrantes:
* Arthur Kafer
* Gabriel Alves
* William Pilger

História (Versões do Kernel)

- 2.4.19 - Surgimento do namespace
  - Criar e gerenciar diversos contextos em um mesmo sistema
  - Início com o **Mount**
- 3.8 - Namespaces de usuário
  - Isolamento de privilégios e identificação de usuários para sets de processos
- 5.6 - Múltiplos tipos de namespace
  - Cada processo é associado com um namespace e pode somente ver ou usar os recursos associados com aquele namespace e seus derivados

Conceito Namespace

- Nome simbólico
  - Partição/isolamento de recursos no kernel
- Consiste em um ou mais processos e um conjunto de recursos
  - Recursos podem ser para múltiplos namespaces
- Porque namespaces são necessários para o Linux?
  - Camada de abstração para os processos
  - Processo só tem conhecimento dos recursos que está utilizando (CPU, memória, etc)

- Docker
  - Ferramenta que manipula namespaces

Tipos de NameSpace

- mnt
- pid
- net
- ipc
- uts
- user
- time
- cgroup

Mount (mnt)

- Partição virtual do sistema de arquivos
- Por padrão, dois namespaces montados em locais diferentes não conseguem ver o conteúdo do outro.
- Na sua concepção, somente o mount root (/) existe, os outros namespaces são todos montados dentro de root.

```
# unshare --mount    # abre um shell em um novo namespace de tipo mount

# mount --bind /usr/bin /mnt/    # linka a pasta /usr/bin (comandos linux) para /mnt/

# ls /mnt/    # lista os mesmos comandos de /usr/bin

# exit    # sai do shell criado, e o mount namespace que foi criado irá sumir

# ls /mnt/   # não irá mais existir o bind feito para /usr/bind no namespace criado (já descartado)
```

Process ID (PID)

- Forma de rastrear processos únicos em um computador
- Todos os processos do Linux são filhos do PID 1, criando a árvore raiz dos processos
  - O namespace do processo isola o novo PID e não permite que ele possa acessar o seu nível superior na árvore
- Processos em diferentes namespaces PID podem ter o mesmo PID

```
# unshare -p -f –mount-proc   # abre um shell em um novo namespace de tipo pid/mount

# top   # Exibe os processos disponíveis dentro daquele namespace

# ls   # lista os diretórios/arquivos. Podemos ver que foram mantidos
```

Network (net)

- Utilizado para gerenciar a parte de rede dentro do namespace
- Cada namespace tem seu endereço IP, rotas de IP, números de porta, etc
- É a partir do namespace de rede que o Docker consegue rodar múltiplos containers internamente
- Ele **não** gerencia sozinho, o usuário terá que criar os dispositivos de rede virtuais e gerenciar as informações da interface pai para as interfaces filho.

```
# ip addr    # lista todos dispositivos configurados de rede
```

Interprocess Communication (ipc)

- Isola comunicação entre processos
  - Cria áreas de memória isolados
  - Inibe comunicação entre processos
  - Restringe funções disponíveis (incluindo chamadas de sistema)
- Múltiplas instâncias de mesma função
  - Uma mesma função/serviço pode existir mutuamente em dois IPCs
- Comunicação entre namespaces IPC distintos
  - É possível vincular IPCs em pontos de montagem distintos

UNIX Time-Sharing (uts)

- Gerencia os hostnames e o domínio NIS (serviço de informações de rede)
- Permite com que o sistema tenha nomes e domínios diferentes para seus processos
- Containers utilizam o namespace UTS para ter um hostname e um domainname diferentes
- Exemplo utilizando o usuário root - repare que a troca não acontece no bash imediato

```
root@vm-ubuntu:/home/arthur# hostname
vm-ubuntu
root@vm-ubuntu:/home/arthur# unshare --uts
root@vm-ubuntu:/home/arthur# hostname
vm-ubuntu
root@vm-ubuntu:/home/arthur# hostnamectl set-hostname teste
root@vm-ubuntu:/home/arthur# hostname
vm-ubuntu
root@vm-ubuntu:/home/arthur# exit
logout
root@vm-ubuntu:/home/arthur# hostname
teste
root@vm-ubuntu:/home/arthur# _

Ubuntu 20.04.4 LTS teste tty1

teste login:
```

User ID (user)

- Permite que o sistema restrinja o acesso a arquivos confidenciais e evita que pessoas que usam o mesmo computador acessem arquivos umas das outras
- Isola os espaços do usuário e seu grupo
- Extremamente importante para segurança do sistema
- Para rastrear as permissões, existe um processo de mapeamento de usuário para um identificador de usuário específico e esse User ID é gravado no arquivo também. Dessa forma, quando é feita alguma alteração no nome do usuário, é simples para o sistema entender quem é o dono de cada arquivo.

Time (time)

- Fornece isolamento para os clocks de boot e possíveis offset para os namespaces
- Usado por containers para configurar datas e horários diferentes do sistema principal
- Virtualiza valores de clock para o namespace
- Pode ser utilizado para criar um processo NTP dentro de container

Control group (cgroup)

- Utilizado para controlar recursos de sistema
- Utilizado também por containers para isolar processos em um sistema virtual no nível de sistema operacional
- Control groups é utilizado por gerenciadores de containers como Docker e tem as seguintes funções:
    - Limitar recursos: definir quotas de um determinado recurso para um processo específico
    - Prioridade: definir prioridade de um recurso de um processo em comparação com outro cgroup
    - Contabilidade: monitorar e relatar os limites de recursos dentro do cgroup
    - Controle: alterar status de todos os processos em um cgroup em um só comando

Referencial

LINUX NAMESPACES, Local: Wikipedia. Disponível em <https://en.wikipedia.org/wiki/Linux_namespaces> acessado em 03/10/2022.

OVENS, Steve. The 7 most used namespaces. Local: RedHat. Disponível em <https://www.redhat.com/sysadmin/7-linux-namespaces> acesso em 03/10/2022.

ACCARDI, Kristen Carlson. Proceedings of the Linux Symposium. Ottawa, Canada. Disponível em <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.108.5475&rep=rep1&type=pdf> Acesso em 02/10/2022.