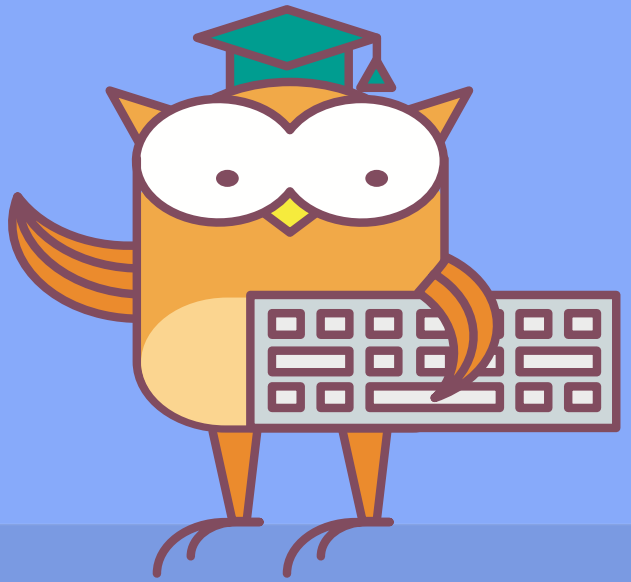
ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

Управление процессами

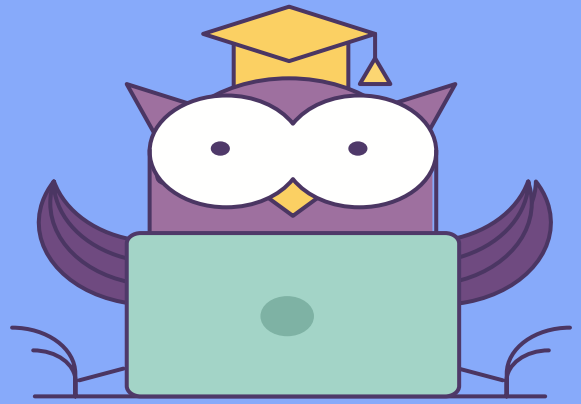
Курс «Администратор Linux»

Занятие № 5



Цыкунов Алексей

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Процессы в linux

- Процесс и его атрибуты
- Жизнь процессов
- Получение информации о процессе

Аттрибуты процесса

- Идентификатор процесса (PID).
- Идентификатор родительского процесса (PPID).
- Идентификатор владельца (UID) и эффективный идентификатор владельца.
- Идентификатор группы GID и эффективный идентификатор группы (EGID)
- GID - это идентификационный номер группы данного процесса. EGID связан с GID также, как EUID с UID.
- PRI - приритет планировщика
- NICE - приоритет планировщика (от -20 - наивысший, до +19 - низший)
- Текущий каталог, корневой каталог, переменные программного окружения,
- Управляющий терминал (controlling terminal)

Примеры ps

1. **Real UserID :** Владелец (тот кто запустил) процесса

2. **Effective UserID :** Обычно равен RUID, но иногда привилегии могут быть повышены через SUID, SGID

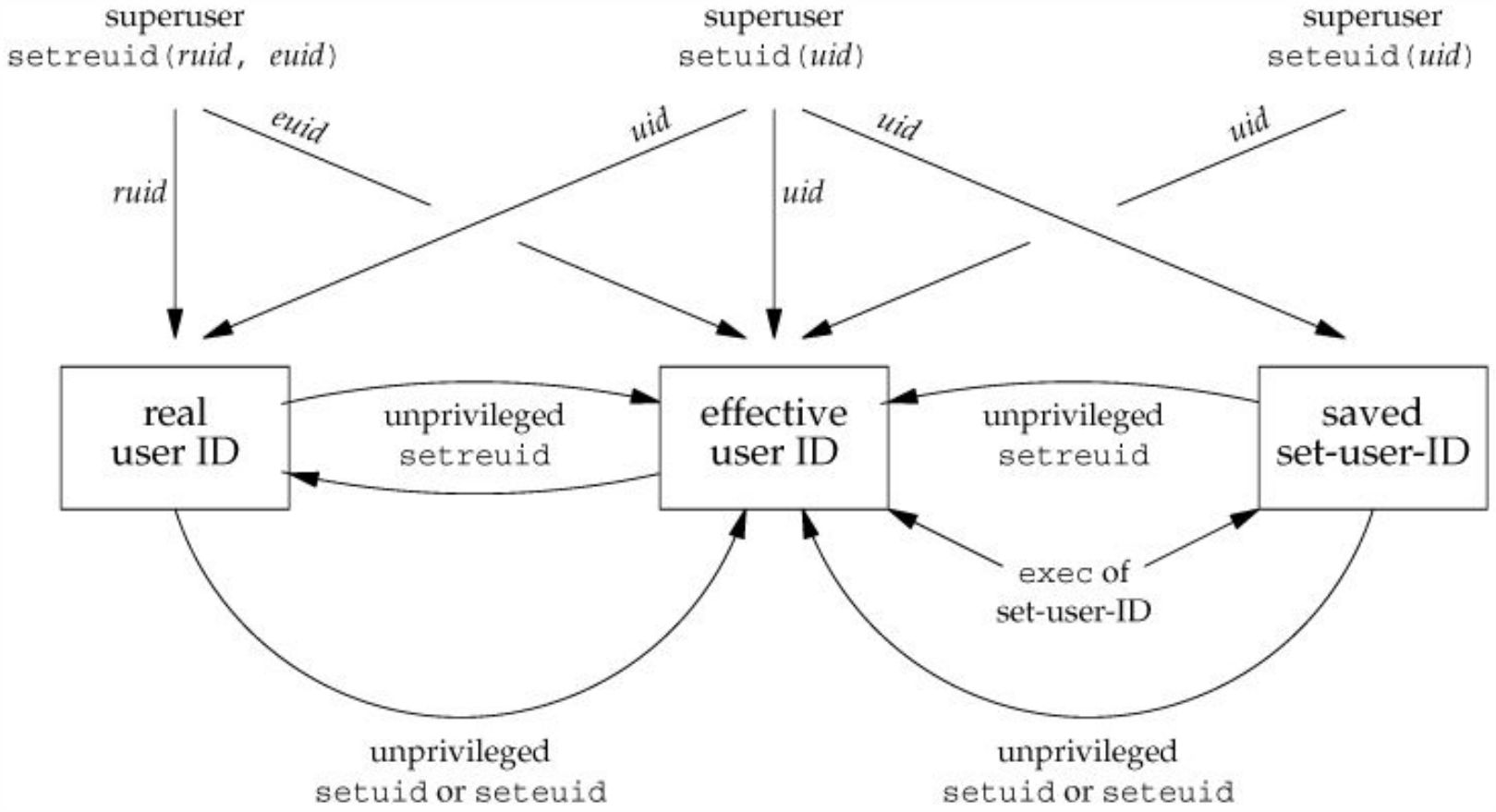
3. **Saved UserID :** Используется при понижении привилегий процесса (обычно процессами запущенными от root). EUID сменяется на ID пользователя с низшим приоритетом а в Saved UID сохраняется предыдущий EUID для переключения обратно

Изменение идентификатора пользователя

OTUS

Идентификатор	**exec**		**setuid(uid)**	
	Бит set-user-ID выключен	**Бит set-user-ID включен**	**Суперпользователь**	**Непривилегированный пользователь**
Реальный	Не изменяется	Не изменяется	Устанавливается в соответствии с *uid*	Не изменяется
Эффективныи	Не изменяется	Устанавливается в соответствии с идентификатором владельца файла программы	Устанавливается в соответствии с *uid*	
Сохраненный	Копия эффективного идентификатора	Копия эффективного идентификатора	Устанавливается в соответствии с *uid*	Не изменяется

Примеры ps


superuser
setreuid(ruid, euid)
superuser
setuid(uid)
superuser
seteuid(uid)
ruid
euid
uid
uid
uid
uid
real
user ID
effective
user ID
saved
set-user-ID
unprivileged
setreuid
unprivileged
setreuid
exec of
set-user-ID
unprivileged
setuid or seteuid
unprivileged
setuid or seteuid

Полезные опции программы ps:

- -ef или ax - расширенный вывод обо всех процессах
- u - информация о пользователе от которого запущен процесс (включено в -ef)
- w - расширить поле cmd (ww - не ограничивать)
- f - вывод дерева процессов
- o - определить формат вывода
- -L - вывести треды

```
[root@linux-demo ~]# ps axfo pid,ppid,pgid,sid,stat,cmd | grep $$
 3375  3372  3375  3375 Ss        \_ -bash
 4025  3375  4025  3375 R+            \_ ps axfo pid,ppid,pgid,sid,stat,cmd
 4026  3375  4025  3375 S+            \_ grep --color=auto 3375
```

Примеры ps

ps -A #Все активные процессы

ps -A -u username #Все активные процессы конекретного пользователя

ps -eF #Полный формат вывода

ps -U root -u root #Все процессы работающие от рута

ps -fG group_name #Все процессы запущенные от группы

ps -fp PID #процессы по PID (можно указать пачкой)

ps -e --forest #Показать древо процессов

ps -fL -C httpd #Вывести все треды конкретного процесса

ps -eo pid,tt,user,fname,tmout,f,wchan #Форматируем вывод

ps -C httpd #Показываем родителя и дочернии процессы

ps -eLf # информация о тредах

ps axo rss | tail -n +2|paste -sd+ | bc

Аттрибуты процесса

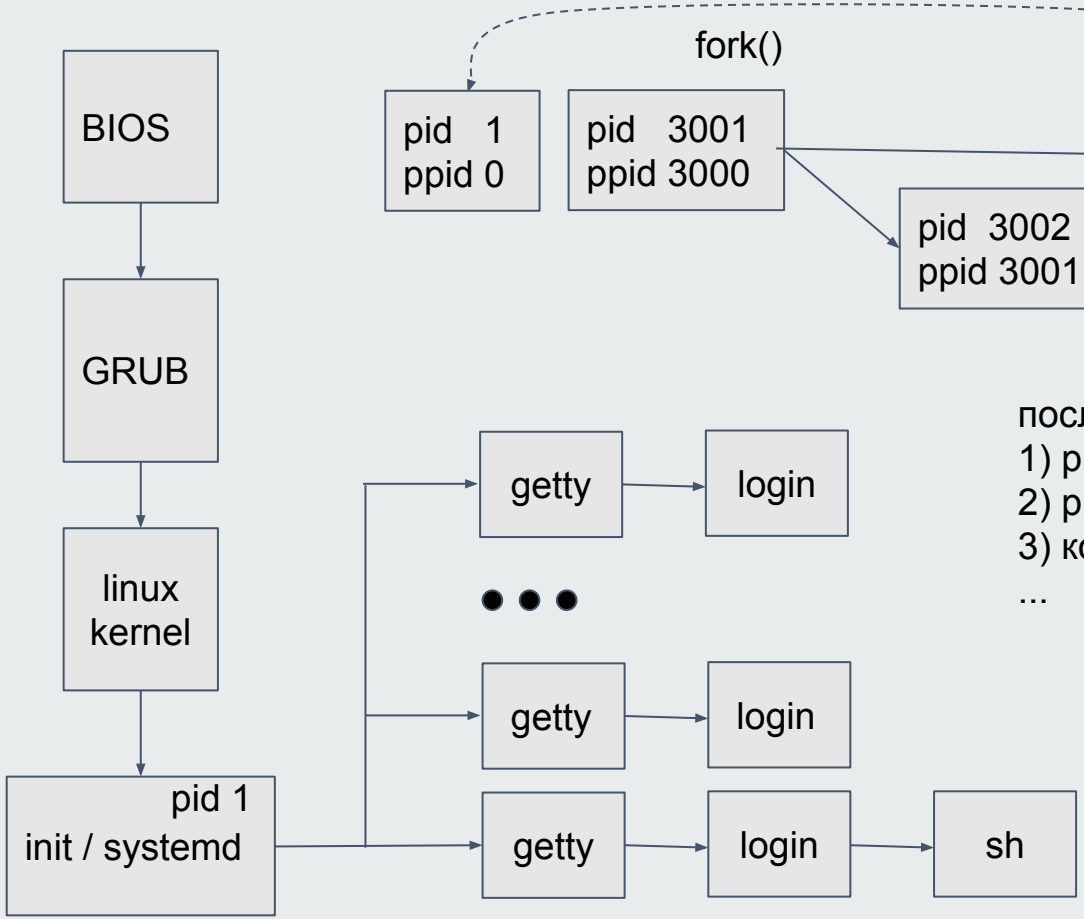
- State
  - R - Running
  - S - Sleeping
  - D - Uninterruptable I/O
  - Z - Zombie
  - t - Trace
  - T - STOP

Аттрибуты процесса

BSD формат (ps ax) или опция state отображает дом символы состояний

- < high-priority (not nice to other users)
- N low-priority (nice to other users)
- L has pages locked into memory (for real-time and custom IO)
- s is a session leader
- l is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
- + is in the foreground process group

Ветвление процессов



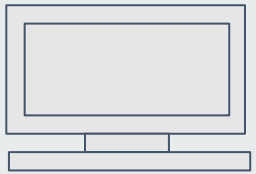
после fork() процессы отличаются:
1) pid
2) ppid
3) код возврата fork()
...

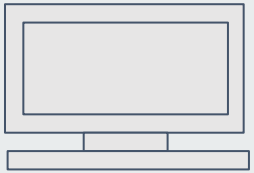
fork()

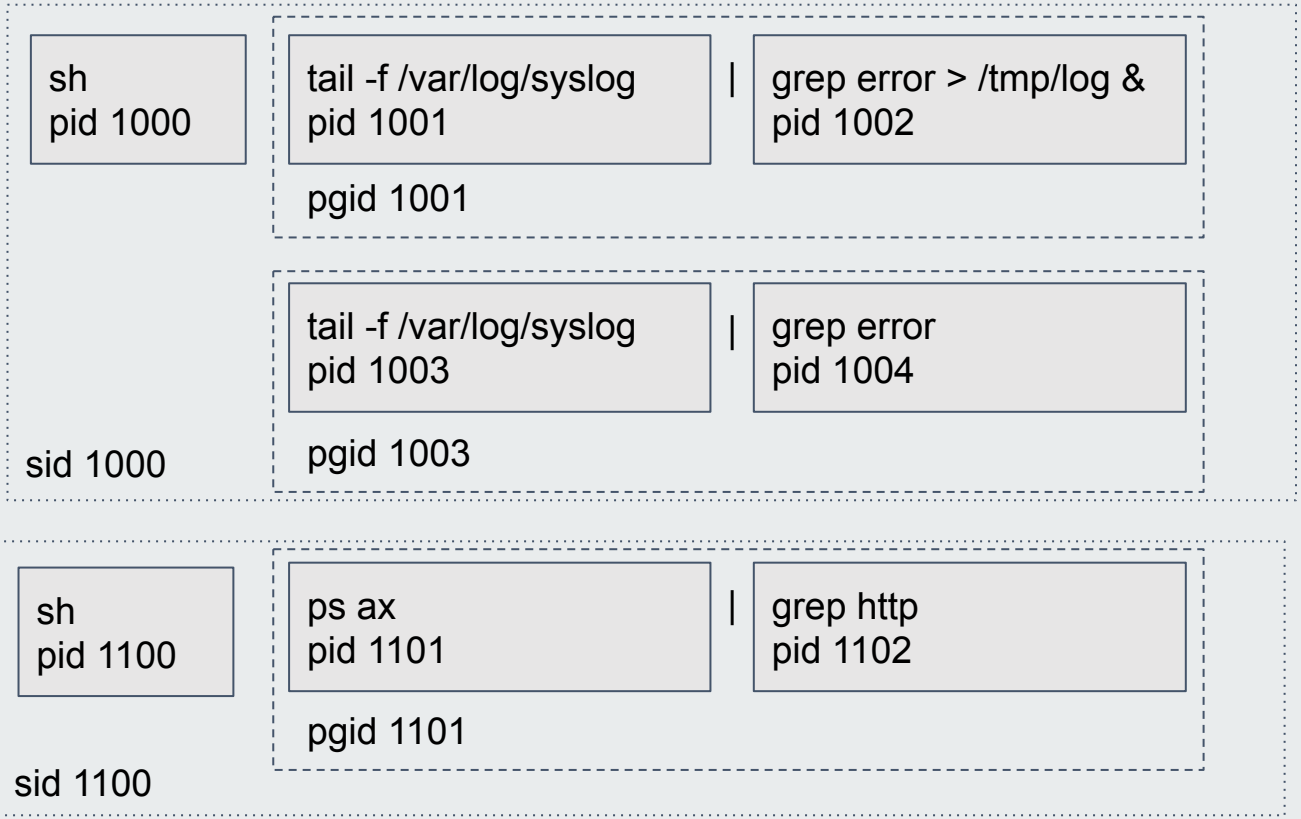
Выполнение fork()

1. Выделяется память для описателя нового процесса в таблице процессов
2. Назначается идентификатор процесса PID
3. Создается логическая копия процесса, который выполняет fork() - полное копирование содержимого виртуальной памяти родительского процесса, копирование составляющих ядерного статического и динамического контекстов процесса-предка
4. Увеличиваются счетчики открытия файлов (порожденный процесс наследует все открытые файлы родительского процесса).
5. Возвращается PID в точку возврата из системного вызова в родительском процессе и 0 - в процессе-потомке.

Ветвление процессов







Интерпретаторы

Любой исполняемый файл в linux запускается через свой интерпретатор/обработчик.

Для интерпретируемых скриптов интерпретатор задается через ‘shebang’ - 1ю строчку в файле начинающуюся с ‘#!’, а далее следует путь к интерпретатору.

Для бинарных файлов elf в качестве обработчика используется /lib64/ld-2.17.so. Это можно использовать для запуска бинарных файлов без атрибута eXecutable.

Процессы и сигналы

Сигналы – это программные прерывания. Сигнал является сообщением, которое система посылает процессу или один процесс посылает другому. Процесс получивший сигнал прерывает свою работу и передает управление обработчику сигнала, По окончанию обработки процесс может продолжить работу

Процессы и сигналы

Сигналы способны в случайное время (асинхронно) прерывать процесс для обработки какого-либо события. Процесс может быть прерван сигналом по инициативе другого процесса или ядра. Ядро использует сигналы для извещения процессов о различных событиях, например о завершении дочернего процесса.

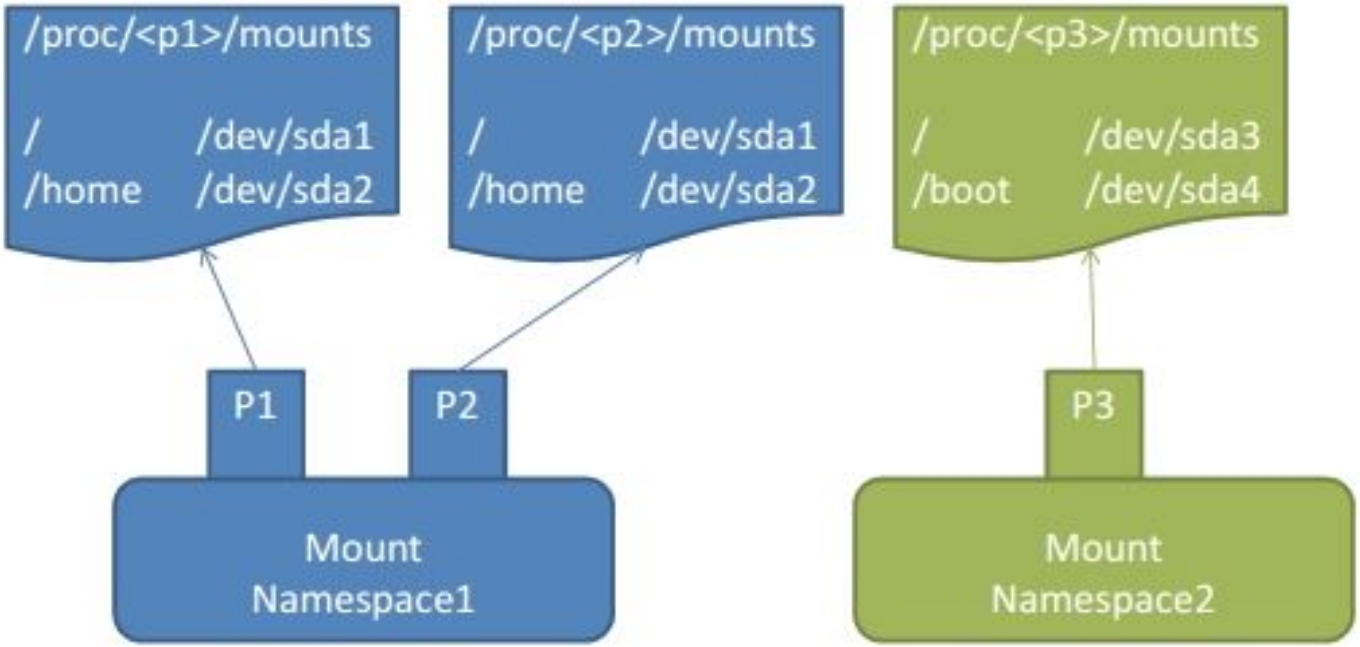
Процессы и сигналы

#	NAME	значение
1	HUP	HangUP - чаще всего перечитать конфигурацию
2	INT	INTerrupt received
9	KILL	незамедлительное завершение процесса
11	SEGV	SEGmentation Violation
13	PIPE	broken PIPE - запись в pipe из которого никто не читает
15	TERM	TERMination signal
19	STOP	STOP Proces

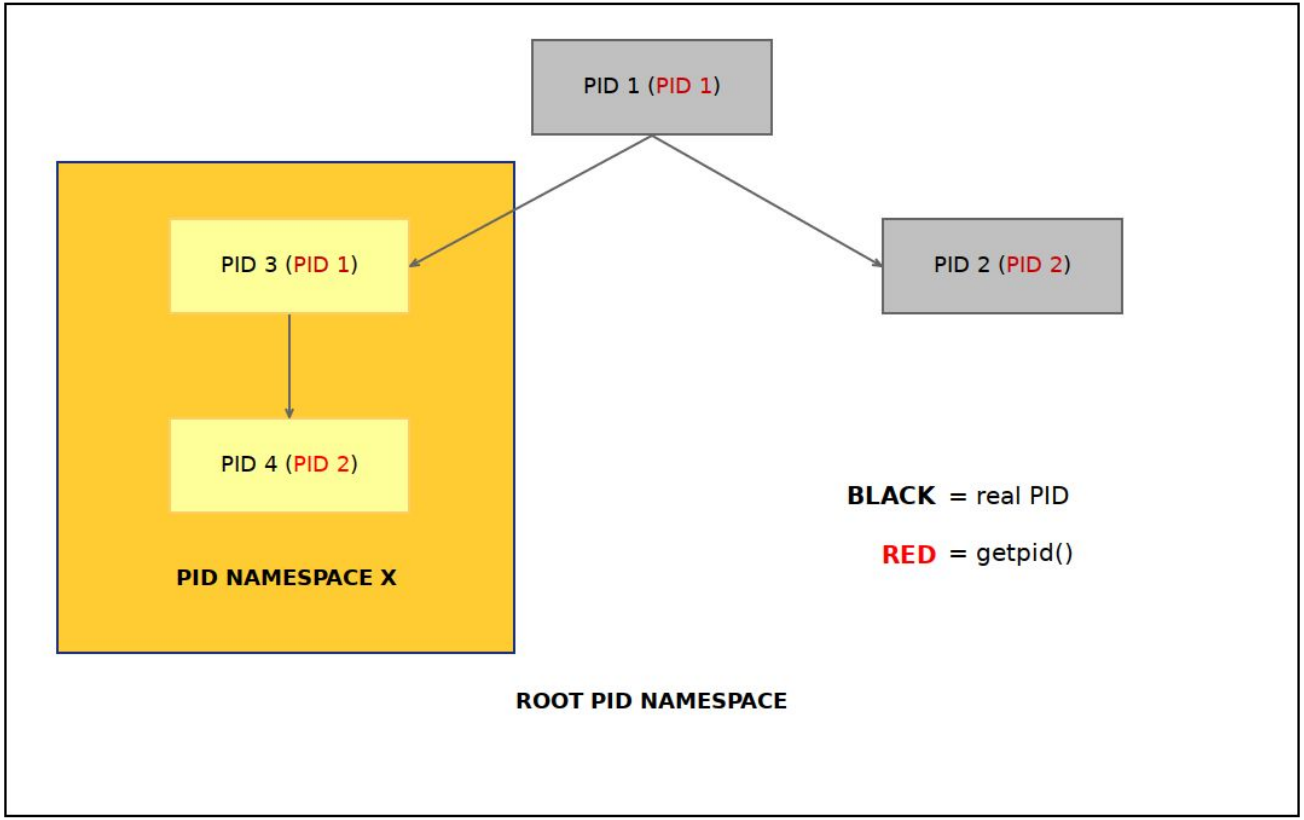
Механизм изоляции структур данных ядра

1. Mount - изоляция точек монтирования
2. UTS - изоляция имени и домена хоста (uname)
3. IPC - изоляция IPC ресурсов
4. PID - изоляция пространства номеров процессов
5. Network - изоляция сетевых интерфейсов
6. User - изоляция пространства идентификаторов UID/GID
7. Cgroup - изоляция cgroup root директории

Namespaces: mount



Namespaces: PID



Namespaces: утилиты

OTUS

unshare - запустить программу в новом пространстве имен (отделить от родительского)

 # unshare --fork --pid --mount-proc readlink /proc/self

$ unshare --map-root-user --user sh -c whoami

nsenter - запустить программу в чужом пространстве имен (войтив чужое пространство)

Получение базовой информации о процессе

- ps
- top
- pstree
- /proc
- lsof
- fuser

pstree -a # Вывод с учетом аргументов командной строки
pstree -c # Разворачиваем дерево еще сильнее
pstree -g # Вывод GID
pstree -n # Сортировка по PID
pstree username # pstree для определенного пользователя
pstree -s PID # pstree для пида, видим только его дерево

lsof

lsof -u username # Все открытые файлы для конкретного пользователя
lsof -i 4 # Все соединение для протокола ipv4
lsof -i TCP:port # Сортировка по протоколу и порту
lsof -p [PID] # Открытые файлы процесса по пиду
lsof -t [file-name] # каким процессом открыт файл
lsof -t /usr/lib/libcom_err.so.2.1 ^^^
lsof +D /usr/lib/locale # Посмотреть кто занимает директорию
lsof -i # Все интернет соединения
lsof -i udp/tcp # Открытые файлы определенного протокола

fuser

fuser -v /mnt # посмотреть все процессы использующие директорию
fuser -v -n tcp 80 # Все процессы использующие 80й порт
fuser -i -k 123/tcp # убить все процессы слушающие порт
fuser -v -m example.txt # найти процессы кто использует файловую систему где расположен файл

procfs: /proc/$pid

- /proc/$pid/
  - cwd - ссылка на текущий каталог процесса
  - cmdline — содержит команду с параметрами с помощью которой был запущен процесс
  - environ — переменные окружения, доступные для процесса
  - maps, statm, и mem — информация о памяти процесса
  - fd/ - директория со списком файловых дескрипторов
  - exe - ссылка на исполняемый файл
  - status - файл с детальной информацией о процессе
  - stat - машиночитаемый файл с информацией о процессе

/proc


```
awk '/32 host/ { print f } {f=$2}' /proc/net/fib_trie | sort | uniq | egrep -v
'127.0.0.1'
/proc/version
/proc/cpuinfo
/proc/meminfo
ls /proc/*/ns/*
cat /proc/sys/net/ipv4/ip_forward
```

Управление фоновыми процессами

- jobs - список запущенных задач.
- & - Выполнить задачу в фоновом режиме.
- Ctrl+Z - Приостановить выполнение текущей (интерактивной) задачи.
- suspend - Приостановить командный процессор.
- fg - Перевести задачу в интерактивный режим выполнения.
- bg - Перевести приостановленную задачу в фоновый режим выполнения.

Расширенная информация о процессе

Помимо того, чтобы наблюдать за процессом со стороны можно заглянуть ему под капот с помощью программ gdb и strace/ltrace.

strace/ltrace

Strace позволяет отследить какие системные вызовы использует процесс.
Для отслеживания вызовов shared библиотек используют ltrace
Полезные опции:

- -c - count statistics
- -s - ограничение длины строковых параметров выводимых на экран
- -t - вывод timestamp вызова
- -T - вывод времени затраченного на вызов
- -f - вывод информации о процессе И его потомках
- -o - вывод информации в файл
- -e - фильтрация вывода

ltrace

ltrace -p <PID> # Дебаг уже запущенного процесса
ltrace -p <PID> -f # Дебаг включая дочернии процессы

gdb

Когда не хватает информации strace можно прибегнуть к более “суровым”/тяжелым методам.

Например, gdb -p pid или gdb /path/to/program /path/to/core и команда bt.

С помощью gdb можно сделать куда более сложные и интересные вещи. Например, поменять лимиты у работающего процесса: https://gchp.ie/updating-ulimit-on-running-linux-process/

GDB: убиваем зомби

OTUS

gdb
Цепляемся к родительскому процессу
attach PID
#Посылаем родидельскому процессу вызов waitpid
call waitpid(PID_zombie,0,0)
wait
detach
quit

Изменение приоритетов процесса

С помощью утилиты nice можно изменить “привлекательность” процесса для планировщика ядра. Большее значение nice уменьшает приоритет, а меньшее - увеличивает.

С помощью утилиты ionice можно изменить приоритет процесса для планировщика ввода-вывода. Есть 3 класса (1: realtime,2: best-effort, 3: idle) приоритетов и 8 уровней приоритета для классов 2 и 3 (меньше уровень - больше приоритет).

Ваши вопросы?

Заполните, пожалуйста, опрос в ЛК о занятии

Спасибо
за внимание!
До встречи в Slack и на вебинаре