РЕФЕРАТ

натему:

”Керування процесами і роботами в Linux”

## Процеси і роботи в Linux

Зміст теми: Керування процесами і роботами в Linux: створення, зупинка, відновлення, завершення. Пріоритети процесів. Активні, фонові та відкладені процеси. Демони.

## Теоретичні відомості

## 1. Основні поняття про процеси і роботи

Процесом у Linux і Unix називається програма або команда, що виконується.

Linux – багатозадачна ОС і в ній одночасно може виконуватись кілька процесів. Для того, щоб їх розрізняти, кожному з них присвоюється персональний ідентифікатор (PID – Process IDentificator). Для виведення на екран списку всіх процесів, що існують в даний момент в системі, використовується команда ps. В результаті виконання цієї команди на екран дисплея може бути, наприклад, видана така інформація:

PID TTY TIME CMD

701 pts/0 1: 16: 00 bash

2403 pts/0 0: 18: 00 bs

TTY – термінал з якого був запущений процес.

ТІМЕ – час, протягом якого він виконується.

CMD – назва програми, яка буде працювати у запущеному процесі.

Найперший процес, запущений системою, називається – init. Його PID=1. Він є головним батьківським процесом всіх інших процесів. Кожен новий процес повинний обов'язково мати свій батьківський процес. Наприклад, процес текстової оболонки bash, який був породжений безпосередньо від процесу init, у свою чергу, буде батьківським процесом для всіх інших процесів, створюваних користувачем під час сеансу. Особливістю Unix та Linux є те, що для кожного нового процесу створюється дублікат батьківського процесу. Такий механізм процесів називається клонуванням.

Кожний новостворений процес отримує три уже відкритих файли:

- stdin – для вхідних даних,

- stdout – для вихідних даних,

- stderr – для повідомлень про помилки.

Процеси можуть функціонувати в двох режимах: системному і користувача. Робота в системному режимі означає виконання процесом системних викликів. Він найбільш важливий, тому що виконується обробка переривань, викликаних зовнішніми сигналами і системними викликами, а також керуванням доступом до диска, розподіл додаткової динамічної пам'яті й інших ресурсів системи. Процес функціонує в режимі користувача, коли виконується програма користувача.

Окрім поняття “процес” використовується також і поняття “робота”. Хоча їх практичний смисл досить близький, проте є деякі відмінності. По-перше, якщо процес характеризується ідентифікатором PID, то робота – номером, що позначається символом %. По-друге, для керування роботами існують спеціальні команди, про які детальніше буде сказано далі. По-третє, до робіт відноситься лише частина наявних в системі процесів, зокрема процеси користувачів.

Варто також зазначити, керування роботами здійснюється власними засобами текстових оболонок Linux. Наприклад, оболонка bash має команду jobs для керування роботами. Для перегляду наявних в системі робіт вказана команда має такий формат:

jobs –<опції>,

де <опції> вказують на ідентифікатори типу процесів (робіт).

Отримати інформацію про відповідність між номером роботи і ідентифікатором процесу можна за допомогою команди

jobs –l

## 2. Активні, фонові та відкладені процеси (роботи)

Процеси (роботи) бувають активними (привілейованими), фоновими та відкладеними. В кожний момент часу може бути лише один активний процес. Активним є такий процес, з яким безпосередньо взаємодіє користувач, тобто тільки цей процес отримує інформацію з клавіатури і посилає результати на ваш екран (як кажуть, виконується на передньому плані). З іншого боку, фонові процеси не одержують інформації з термінала, у загальному випадку вони спокійно виконуються, не вимагаючи потреби в спілкуванні з користувачем. Деякі фонові процеси виконуються протягом великого проміжку часу і не здійснюють нічого зовні цікавого. Компіляція програм або ущільнення файлів - приклади таких процесів. Немає потреби чекати, коли ці процеси закінчаться. Їх можна просто запустити у фоні. Поки вони там виконуються, ви можете займатися іншими програмами. Але потрібно знати основні особливості фонової обробки:

фоновий процес не допускає введення з клавіатури;

будь-яке виведення від фонового процесу на екран руйнує все, що ви в цей момент ввели з клавіатури;

при запуску великої кількості фонових процесів можна перевантажити систему.

Процеси можуть бути також відкладені. Відкладений процес - це процес, що у даний момент не виконується і тимчасово зупинений. Після того, як ви призупинили виконання процесу, надалі ви можете його продовжити як на передньому плані, так і в фоні. Поновлення призупиненого процесу не змінить його стану - при поновленні він почнеться з того місця, на якому сталась зупинка.

Майте на увазі, що призупинення процесу - це не переривання процесу. Коли ви перериваєте процес, який виконується, натискаючи клавіші переривання (зазвичай, це <CTRL><C>), то процес знищується назавжди. А якщо процес знищено, то немає іншого способу відновити його, як знову запустити спочатку, використовуючи колишню команду. Зауважимо також, що деякі програми можуть перехоплювати переривання, тоді натискання клавіш <CTRL><C> не приведе до негайного припинення процесу. Це дозволить програмі виконати необхідні операції акуратного завершення. Деякі програми взагалі не дозволять вам їх перервати.

## 3. Створення процесу

Процес породжується за допомогою системного виклику fork(). При цьому виклику відбувається перевірка на наявність вільної пам'яті, доступної для розміщення нового процесу. Якщо необхідна пам'ять доступна, то створюється процес-нащадок поточного процесу, що представляє собою точну копію процесу, що викликається При цьому в таблиці процесів для нового процесу будується відповідна структура. Нова структура створюється також у таблиці користувача. При цьому всі змінні ініціалізуються нулями. Цьому процесу присвоюється новий унікальний ідентифікатор, а ідентифікатор батьківського процесу запам'ятовується в блоці керування процесом.

Процеси, що виконують різні програми, утворюються завдяки застосуванню наявних у стандартній бібліотеці Linux функцій “сімейства ехес”: ехесl, ехесlр, ехесlе, ехесv, ехесvе, ехесvр. Ці функції відрізняються форматом виклику, але в остаточному підсумку виконують одну задачу: заміщають всередині поточного процесу код, що виконується, на код, що міститься в зазначеному файлі. Цей файл може бути не тільки двійковим файлом Linux, що виконується (аналог файла \*. exe в системі Windows), але і сценарієм командного інтерпретатора, і двійковим файлом іншого формату (наприклад, класом java, що виконується файлом DOS).

Таким чином, якщо операція запуску програми у DOS і Windows виконується як єдине ціле, то у Linux (і в Unix взагалі) розділена на два етапи: спочатку здійснюється запуск, а потім визначається, яка програма буде працювати. Чи є в цьому сенс і чи не занадто великі додаткові витрати? Адже створення копії процесу передбачає копіювання дуже значного обсягу інформації.

Сенс у даному підході є суттєвий. Дуже часто програма повинна зробити деякі дії ще до того, як почнеться власне її виконання. Наприклад, створити непоіменований канал для спілкування з іншими процесами. Реалізується це дуже просто: спочатку “відмежовуються” процеси, потім виконується системний виклик рiре() для створення каналу - і тільки після цього запускається програма із функції еxес().

Стосовно додаткових витрат, то вони найчастіше виявляються надто малими: при створенні копії процесу його індивідуальні дані фізично нікуди не копіюються. Замість цього використовується техніка, відома за назвою сору-оn-write (копіювання при записі): сторінки даних обох процесів особливим чином позначаються, і тільки тоді, коли один процес намагається змінити вміст будь-якої своєї сторінки, він дублюється. Щоб створити новий активний процес рядовому користувачеві зовсім не обов’язково знати про всі ці системні виклики. Йому достатньо лише ініціювати виконання команди або програми, набравши в командному рядку відповідне ім’я файла, або натиснувши на відповідну піктограму на робочому столі. Приклад запуску годинника:

xclock

Для запуску процесу у фоновому режимі необхідно після імені команди чи імені файла ввести символ & і лише потім натиснути на клавішу <Enter>. Приклад запуску годинника в фоновому режимі:

xclock &

Щоб призупинити активний процес необхідно одночасно натиснути клавіші <Ctrl><Z>. Поки процес зупинено на нього не витрачається час процесора. Але ви завжди можете відновити процес, причому як на передньому плані, так і в фоні. Для поновлення процесу в активному режимі використовується команда fg (“foreground” - передній план), а в фоновому режимі – команда bg (“background” - задній план).

Ще одне зауваження. Команди fg і bg звичайно переводять на передній план чи у фоновий режим процеси, що були зупинені останніми (що визначається символом “+” після номера роботи при використанні команди jobs).

Якщо ви виконуєте багато процесів (робіт) одночасно, ви можете перевести на передній план чи, навпаки, у фоновий режим конкретну роботу із заданням її номера як аргумента команд fg або bg:

fg%2 (переведення на передній план роботи номер 2),

bg%3 (переведення у фон роботи номер 3).

Для цих команд не можна використовувати ідентифікатори процесів. Крім того, використання тільки номерів робіт, наприклад

%2

еквівалентно команді

fg%2

Тип процесу (роботи) можна легко визначити за допомогою команди jobs. Наприклад, якщо після виконання цієї команди ми отримали на екрані дисплея інформацію

[1] - Running xclock

[2] - Running xeyes &

[3] + Stopped xcalc

то це означає, що перший процес (робота) в даний момент є активним, другий процес (робота) – фоновим, а третій процес (робота) – відкладений.

Можна також отримати список процесів (робіт) лише одного типу, вказавши відповідну опцію в команді jobs:

jobs –r - видача лише активних процесів (робіт),

jobs –s - видача лише відкладених процесів (робіт).

Пам’ятайте, що керування роботами, це властивість оболонки операційної системи. Команди fg, bg і jobs - внутрішні команди оболонки bash. Якщо з якоїсь причини ви використовуєте текстову оболонку, що не підтримує керування роботами, там ви не знайдете цих команд. На додаток до цього, є деякі аспекти керування роботами, що розрізняються в оболонках bash і tcsh. Деякі оболонки не здатні керувати роботами, хоча більшість оболонок Linux мають таку можливість.

## 4. Пріоритети процесів

Зазвичай при запуску всі процеси користувачів мають однаковий пріоритет, рівний 10, що дає змогу операційній системі рівномірно розподіляти між ними процесорний час. Пріоритет можна змінити за допомогою команди

nice –число процес,

де число – величина, на яку зменшується початкове значення пріоритету для процесу.

Пріоритет активного процесу з ідентифікатором PID можна змінити командою

renice –число PID.

Команда nice не завжди знижує пріоритет.д.ля його підвищення необхідно вказати від’ємне число.

Варто відзначити, що лише суперкористувач має таке право.

## 5. Завершення процесів

Для примусового завершення активного і фонових процесів використовуються різні способи. Як вже раніше відмічалось, активний процес можна ліквідувати, натиснувши клавіші <CTRL><C> або клавішу DEL.

Для завершення фонового процесу використовується команда kill, яка має кілька форматів:

kill PID

kill - signal PID

kill%n

Ця команда може брати як аргумент номер роботи, або ідентифікатор процесу. Наприклад, для завершення процесу із ідентифікатором PID=237 необхідно виконати команду

kill 237,

а для завершення роботи із номером 20 необхідно виконати команду

kill%20

Для перевірки ліквідації вказаного процесу, можна виконати команду

ps

в результаті чого на екрані дисплея отримаємо відповідь:

237 Terminated

А якщо виконати команду

jobs

тоді теж одержимо аналогічне підтвердження:

[20] + Terminated

Ключ “-signal” змушує команду kill виконати ряд додаткових послуг, тобто послати процесу певний сигнал. Може бути послано понад 20 сигналів, кожний з яких має свій номер.

При виході користувача із системи, Linux посилає всім його процесам сигнал 1, що змушує всі процеси завершити роботу. За замовчуванням усім процесам посилається сигнал 15.

Якщо ввести команду kill 0, то можна ліквідувати всі фонові процеси.

Якщо який-небудь процес “завис”, тоді потрібно перейти до іншої консолі, і з її допомогою ввести команду kill для ліквідації “завислого” процесу.

Гарантовано можна знищити процес за допомогою сигналу 9, наприклад:

kill - 9 125

Звичайний користувач має право припиняти тільки процеси, запущені з його термінала. Для завершення процесу використовується системний виклик ехit(), при якому звільняються усі використовувані ресурси, зокрема такі, як пам'ять і структури таблиць ядра. Крім того, завершуються і процеси-нащадки, породжені даним процесом.

Потім з пам'яті вилучаються сегменти коду і даних, а сам процес переходить у стан “зомбі” (у полі Stat такі процеси позначаються буквою “Z”). “Зомбі” не займає процесорного часу, але рядок у таблиці процесів залишається, і відповідні структури ядра не звільняються.

Якщо батьківський процес з якоїсь причини завершиться раніше дочірнього, останній стає “сиротою” (orphaned process). “Осиротілий” “зомбі” на короткий час стає нащадком init, після чого вже остаточно “помирає”.

Також, процес може впасти в “сон”, що не вдається перервати (у полі Stat це позначається буквою “D”). Процес, що знаходиться в такому стані, не реагує на системні запити і може бути знищений тільки перезавантаженням системи.

## 6. Корисна інформація про фонові процеси

Коли текстова оболонка типу bash завершує роботу, вона посилає в усі породжені нею процеси сигнал “відбій”. Якщо процес виконується у фоновому режимі, цей сигнал часто знищує його, що в більшості випадків небажано. Якщо Ви маєте намір запустити у фоновому режимі програму, яка повинна буде працювати і навіть після завершення батьківського процесу (звичайні фонові процеси при цьому завершуються), її потрібно запускати як глибокий фоновий процес командою nohup. Така команда має формат:

nohup команда &

Подібний запуск змушує зазначену аргументом команду ігнорувати сигнал відбою. Команда nohup має і побічний ефект: до значення nice додається п’ять. Уся вихідна інформація, яка генерує процес, якщо стандартний файл виведення і стандартний файл помилок не перепризначені, розміщуються у файлі nohup. out.

Демон (від англійського demon чи daemon) - це фоновий процес, що виконує системну задачу, непомітно для користувача і доповнює операційну систему будь-яким спеціальним сервісом. Ця програма не викликається користувачем у явній формі, а спокійно очікує в пам'яті певної події. У повній відповідності з пануючим у UNIX принципом модульності демони є програмами, а не частинами ядра. Багато демонів запускаються під час початкового завантаження і продовжують працювати увесь час, поки система включена. Інші демони запускаються при необхідності і працюють стільки, скільки передбачено їх функціями.

Демони можна знайти за допомогою команди

ps - ax

До основних демонів можна віднести init, inetd та cron.

Демон init - це перший процес, що запускається після початкового завантаження системи, і є предком майже всіх процесів, його PID завжди дорівнює 1),

Демон inetd керує іншими демонами. Раніше всі демони запускалися під час початкового завантаження операційної системи, і працювали безупинно (точніше, блокувалися при чеканні роботи). Згодом у систему вводилися все нові і нові демони. Їх стало стільки багато, що почали з’являтися проблеми з продуктивністю роботи системи. У відповідь фахівці BSD розробили inetd - демон відповідальний за запуск інших демонів по необхідності. Він запускає демони-клієнти, коли для них є робота, а після виконання задачі дозволяє їм тихо завершитись. Для того щоб працювати під керуванням inetd, клієнти повинні дотримуватись особливих правил: якщо конфігурація операційної системи із самого початку не передбачала використання inetd, то для його введення в систему необхідно модифікувати багато інших програм. Багато демонів можуть використовуватися або традиційним способом (тобто вони запускаються однократно і працюють до вимикання системи), або під контролем inetd.

Демон cron відповідає за виконання команд за графіком. Він обробляє файли з розкладом задач (cron-файли), створені як користувачами, так і адміністратором. Демон cron часто використовують в адміністративних задачах, таких, як керування обліковими файлами і файлами реєстрації, щоденне чищення файлової системи.

## Використана література

1. Топхем Д., Чыонг Х.В. Юникс и Ксеникс. - М.: Мир, 1988. - 392 с.

2. Скловская С. Red Hat Linux 6.0 – Учебник. К.: “ДиаСофтЮП”, 1999. - 416 с.

3. Скловская С. Команды Linux: Справочник. - К.: “ДиаСофт”, 2001. - 688 с.

4. Глушаков С.В., Сурядный А.С. Linux для дома и офиса: Учебный курс. - Харьков: Фолио, 2002. – 389 с.