Процеси

9. Процеси  
• Същност и жизнен цикъл на процес;  
• Преглед на процесите ( ps, pstree, pgrep, top, jobs);  
• Сигнали и тяхното изпращане ( kill, killall, pkill);  
• Управление на задачи (fg, bg);  
• Приоритетизация (nice, renice).

В живота на обикновения ГНУ/Линукс потребител често се среща термина  «процеси». Така че какво точно е «процес»?  Ще опитаме да разберем.  
  
Сухата формулировка ни говори, че процес – е съвкупност от програмен код и данни заредени в паметта на IBM. На пръв поглед процес – това е стартирана програма(приложение) или команда. Но това не е съвсем така – някои  приложения могат да създават няколко процеса едновременно например.  
  
Не е задължително кода на процеса да се изпълнява в момента, тъй като процеса може да се намира в заспало състояние. В този случай изпълнението на кода на такъв процес е спряно. Съществуват всичко на всичко 3 (три) състояния, в които може да се намира процеса:  
  
Работещ процес – в дадения момент кода се изпълнява.  
  
Спящ процес – в дадения момент, кода на процеса не се изпълнява и очаква някакво събитие(например натискане на клавиш, вход на данни  или други).  
  
Процес-зомби – самия процес на съществува, неговите данни са премахнати от оперативната памет но записа в таблицата на процесите остава поради различни причини.  
  
Всеки процес в системата се обозначава с цифров идентификатор (личен номер PID) в диапазона от 1 до 65535(PID – Process Identifier – идентификатор на процеса) и идентификатор на родителския процес (PPID – Parent Process Identifier – идентификатор на родителския  процес). PID се явява името на процеса, по който можем да адресираме процеса в операционната система при използването на различни средства за преглеждане и управление на процесите. PPID определя родствените отношения между процесите, които в значителна степен определят неговите свойства и възможности. Другите параметри, които са необходими за работа на програмата се наричат”обкръжение на процеса”. Един от тези се явява управляващия терминал – име на терминалното устройство на което процеса извежда информация и от където получава информация. Управляващия терминал не управлява обаче всички процеси. Съществуват такива, които не са привързани към никой терминал и се наричат “демони” (daemons). Такива процеси са стартирани от потребителя и не завършват своята работа при завършване на сеанса, а продължават да работят, тъй като те не са свързани с текущия сеанс и не могат да бъдат автоматично завършени. Като правило, с помощта на демоните се реализират сървърните служби  като например сървъра за печат е реализиран на демона cupsd, а сървъра за журналните файлове  – syslogd.  
  
За разглеждане списъка на процеси в ГНУ/Линукс съществува команда  ps. Формата и е следния:  
  
ps [PID] [options] –   
  
преглежда списъка на процеси. Без параметри ps показва всички процеси, които са били стартирани в течение на текущата сесия, с изключение на демоните. Options може да приема едно от следните значения или техните комбинации:  
  
-а или -e – показва всички процеси  
  
-f – пълен листинг  
  
-w – показва пълните редове описващи процесите. Ако редовете са по дълги от широчината на екрана, пренася реда в следващия ред.  
  
Това не са всичките параметри на ps. Останалите може да разберете чрез man ps.  
  
Пример1:  
  
[gserg@WEBMEDIA gserg]$ ps  
  
PID TTY TIME CMD  
  
3126 pts/2 00:00:00 bash  
  
3158 pts/2 00:00:00 ps  
  
[gserg@WEBMEDIA gserg]$\_  
  
  
  
Пример2:  
  
[gserg@WEBMEDIA gserg]$ ps 3126  
  
PID TTY STAT TIME COMMAND  
  
3126 pts/2 S 0:00 /bin/bash  
  
[gserg@WEBMEDIA gserg]$\_  
  
  
  
Пример3:  
  
[gserg@WEBMEDIA gserg]$ ps -ef  
  
UID PID PPID C STIME TTY TIME CMD  
  
root 1 0 0 10:01 ? 00:00:03 init [5]  
  
root 2 1 0 10:01 ? 00:00:00 [keventd]  
  
root 3 1 0 10:01 ? 00:00:00 [kapmd]  
  
root 4 1 0 10:01 ? 00:00:00 [ksoftirqd\_CPU0]  
  
root 5 1 0 10:01 ? 00:00:24 [kswapd]  
  
root 6 1 0 10:01 ? 00:00:00 [bdflush]  
  
…  
  
gserg 3126 3124 0 17:56 pts/2 00:00:00 /bin/bash  
  
gserg 3160 3126 0 17:59 pts/2 00:00:00 ps -ef  
  
[gserg@WEBMEDIA gserg]$\_  
  
  
  
Пример4:  
  
[gserg@WEBMEDIA gserg]$ ps -efw  
  
UID PID PPID C STIME TTY TIME CMD  
  
root 1 0 0 10:01 ? 00:00:03 init [5]  
  
root 2 1 0 10:01 ? 00:00:00 [keventd]  
  
root 3 1 0 10:01 ? 00:00:00 [kapmd]  
  
root 4 1 0 10:01 ? 00:00:00 [ksoftirqd\_CPU0]  
  
root 5 1 0 10:01 ? 00:00:24 [kswapd]  
  
……  
  
root 1130 1 0 10:02 ? 00:00:00 /usr/sbin/apmd -p 10 -w 5 -W -P /etc/sysconfig/apm-scripts/apmd\_proxy  
  
gserg 3172 3126 0 18:01 pts/2 00:00:00 ps -efw  
  
[gserg@WEBMEDIA gserg]$\_  
  
  
  
Процесите в ГНУ/Линукс притежават същите права както и потребителите, които ги изпълняват..  
  
В крайна сметка операционната система възприема работещите потребители като набор от стартирани от него процеси. Така и самия сеанс се открива в команден ред (или Х обвивка) от името на потребителя. Затова когато говорим за „права на достъп за файла” то трябва да се разбира „права на достъп на стартираните от потребителя процеси към файла”.  
  
За определяне името на потребителя, на заспиващия процес операционната система използва реални идентификатори на потребителя и групите, назначаващи процеса. Но тези идентификатори не се явяват решаващи при определяне правата на достъп. За това при всеки процес съществуват група идентификатори наречени  - ефективни..  
  
Как правило,  реалните и ефективните индикатори на процесите са еднакви, но има и изключения. Например, за работата на инструмента passwd е необходимо да се използва идентификатор за супер потребител, тъй като само супер потребителя има право да записва във файла с пароли. В този случай ефективните идентификатори на процеса ще се отличават от реалните. Възниква резонния въпрос – как е реализирано това?  
  
Всеки файл има набор от специални праза за достъп – битове SUID и SGID. Тези битове позволяват при стартирането на програмата да и бъдат присвоени ефективни идентификатори на потребителя и на групата на потребителя съответно и да се изпълнява процес с права на достъп с друг потребител. Така както passwd принадлежи на root потребителя и той е с настроен бит SUID, то при стартирането на процеса passwd ще бъде с права на root.  
  
Установяване на SGID и SUID чрез командата chmod:  
  
chmod u+s filename – задава бита SUID  
  
chmod u+s filename – задава  бита SGID   
  
До тук разгледахме понятието процес, начини за изобразяване на процеса и права на достъп. Но за комфортната работа в операционната система, това е наистина малко. Необходимо е още ефективно да се управляват процесите. А за реализацията на управлението трябва да прегледаме като начало таблицата на процесите:  
  
Родител на всички процеси в системата е процеса init. Неговия PID винаги е 1, PPID – 0. Цялата таблица на процесите може да се представи във вид на дърво, в чиято основа ще бъде процеса init. Този процес не е част от ядрото, но изпълнява много важна роля – определя текущото равнище на инициализация на системата и следи за стартирани програми, позволяващи на потребителя да работи с компютъра (mingetty, X или други).  
  
Процеси, имената на които започват в квадратни скоби – например “[keventd]” – са процеси на ядрото. Тези процеси управляват работата на системата и по-точно нейните части, като мениджъра на паметта, мениджъра на порцесорните времена, мениджъра на външните устройства и така нататък.  
  
Останалите процеси, се явяват потребителски, стартирани обикновено от конзолен ред, или пък по време на инициализацията на системата.  
  
Живота на всеки процес е представен със следните фрази:   
  
Създаване – на този етап се създава пълно копие на този процес, който създава нов. Например, стартирате от интерпретатора командата ls. Командния интерпретатор създава свое собствено пълно копие на тази команда.  
  
Зареждане кода на процеса и подготовка към стартиране – копието на създадената програма на първия етап, заменя кода на задачата, която е необходимо да изпълни и създава нейното обкръжение – установява необходимите променливи и т.н.  
  
Изпълнение на процеса   
  
Състояние зомби – на този етап от изпълнението, процеса е завършен, кода е изтрит от паметта, обкръжението се унищожава но записа в таблицата на процесите още стои.  
  
Умиране на процеса – след всички завършващи стадии се изтрива записа в таблицата на процесите и самия процес завършва своята работа.  
  
По време на работата на процесите, ядрото контролира тяхното състояние и в случай на възникване на непредвидени ситуации управлява процеса с помощта на изпращане на сигнал. Сигнал – това е прост между процесорен способ(има се предвид между процесите) за взаимодействие. Съществуват няколко типа сигнали. За всеки от типовете е предвидено действие по подразбиране. Процеса може да се използва действието по подразбиране или ако в него има обработващ сигнала модул то той може да го прихване и обработи или игнорира. Сигналите SIGKILL и SIGSTOP е невъзможно да се игнорират.  
  
По подразбиране са възможни няколко действия:  
  
Игнориране – продължава работата, въпреки получения сигнал.  
  
Завършване – процеса спира работа.  
  
Завършване + core – завършва работата на процеса и създава файл в текущата директория, съдържащ образа на процеса от паметта (код и данни).  
  
Спиране – преустановява изпълнението на процеса но не спира работата му и не го изтрива от паметта.  
  
Ето списък със сигналите, съществуващи в системата към момента:   
  
Име  
Действие по подразбиране  
Значение  
  
SIGABRT  
Завършва + core  
Сигнал се изпраща ако самия процес извиква  abort()  
  
SIGTERM  
Завършва  
Сигнала обикновено представлява своего рода  предупреждение, че процеса в най-скоро време ще бъде унищожен. Този сигнал позволява на процеса  един вид да се „подготви за смъртта” т.е. да изтрие временните файлове, завърши необходимите прехвърляния и т.н. Командата kill по подразбиране изпраща именно този сигнал .  
  
SIGTTIN  
Спира  
Сигнала се генерира от ядрото(драйверите на управляващия терминал), при извикване на процеса, фонова група осъществява четене от управляващия терминал.   
  
SIGTTOU  
Спира  
Сигнала се генерира от ядрото (драйверите на терминала) при извикване на процеса фонова група осъществява запис на управляващия терминал.  
  
SIGALRM  
Завършва  
Сигнала се изпраща, когато е зададен по-рано таймер.  
  
SIGBUS  
Завършва + core  
Сигнала свидетелства за някои апаратни грешки. Обикновено този сигнал се изпраща при обръщения към недопустим адрес във виртуалната памет, за който липсва съответна физическа страница.   
  
SIGCHLD  
Игнорира   
Сигнал, изпращан от родителския процес при завършване на неговия потомък.  
  
SIGSEGV  
Завършва + core  
Сигнала свидетелства за обръщение на процеса към недопустима област в паметта, за която в процеса няма привилегии за достъп.  
  
SIGFPE  
Завършва + core  
Сигнала свидетелства за възникване на особена ситуация, като деление на  0 или изпълняване на операции с плаваща точка.  
  
SIGHUP  
Завършва  
Сигнала се изпраща към лидера на сеанса, свързан с управляващия терминал, за това че терминала се е изключил (пропадане на връзката). Сигнала също може да бъде изпратен до всички процеси в текущата група при завършването на изпълнението от лидера.   
  
Този сигнал понякога използва в качеството на просто средство междупроцесното взаимодействие. В частност, той се приема за съобщение демон за необходимостта от обновяване на конфигурационната информация. Причина за избор именно на този сигнал – SIGHUP , е че демона по принцип няма управляващ терминал и обикновено не получава този сигнал.  
  
SIGILL  
Завършва + core  
Сигнала се изпраща от ядрото, ако процеса се опита да изпълни недопустима инструкция.  
  
SIGINT  
Завършва  
Сигнала се изпраща от ядрото до всички процеси при натискане на клавиша (<CTRL>+<C>)  
  
SIGKILL  
Завършва  
Сигнал, при получаването при получаването на който изпълнението на процеса  се прекратява. Този сигнал не може да се прихване и игнорира.   
  
SIGPIPE  
Завършва  
Сигнала се изпраща при опит за запис в сокет, получателя на данни на който е завършил изпълнението или е закрил файловия указател на сокета.  
  
SIGPOLL  
Завършва  
Сигнала се изпраща при настъпване на определено събитие за устройството, което се явява отговорно(например получаване на пакет по мрежата).  
  
SIGPWR  
Игнориране  
Сигнала се генерира при заплаха от загуба на  захранване. Обикновено се изпраща,  когато системата се превключва на UPS.   
  
SIGQUIT  
Завършва  
Сигнала се изпраща към всички процеси от текущата група при натискане на клавиш <CTRL>+<\>.  
  
SIGSTOP  
Спира  
Сигнала се изпраща до всички процеси в текущата група при натискане на клавиши <CTRL>+<Z>. Получаването на този сигнал извиква спиране на процеса.  
  
SIGSYS  
Завършва + core  
Сигнала се изпраща от ядрото при опит да се изпълни процес  с недопустим системен сигнал.  
  
SIGUSR1  
Завършва  
Сигнала е предназначен за допълнителни задачи като просто средство за междупроцесно взаимодействие.  
  
SIGUSR2  
Завършва  
Сигнала е предназначен за допълнителни задачи като просто средство за междупроцесно взаимодействие.  
  
  
Немаловажна роля в живота на процесите играе мениджъра на процесите – това е част от ядрото, отговаряща за многозадачността на системата. Ясно е че за единица време в един процесор може да се изпълнява само един процес. Именно този мениджър определя, кой от стартираните процеси първи да се изпълнява и кой втори.  За да можем да видим този приоритет на процесите трябва да изпълним ps с опция – l(long – разширен изход):  
  
[gserg@WebMedia gserg]$ ps -l  
  
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD  
  
0 S 500 1554 1553 0 75 0 - 1135 wait4 pts/1 00:00:00 bash  
  
0 R 500 1648 1554 0 81 0 - 794 - pts/1 00:00:00 ps  
  
[gserg@WebMedia gserg]$  
  
По време на своята работа мениджъра на процеси в началото поставя за изпълнение процеси с нисък приоритет.  Така приоритет 0 е зает само за критичните системни задачи, а отрицателния приоритет за процеси на ядрото. Задачи с голям приоритет изискват малко процесорно време и затова като правило работят бавно и губят малко системни ресурси.  
  
Остава само да решим въпроса, може ли потребителя да управлява процесите и системните параметри ? Естествено, че може! За това в ГНУ/Линукс има набор от инструменти позволяващи промяна приоритета на процеса, чрез изпращане на сигнал. За тях става въпрос по-долу.  
  
Първия инструмент – командата nice:  
  
nice -n command – позволява да се промени приоритета, с който ще се изпълнява процеса след стартирането. Без указания командата дава текущия приоритет на процеса и по подразбиране той е равен на 10. Диапазона на приоритетите е разположен от -20(най – старши ) до 19(най – низш). Пример е показан по – долу:  
  
[gserg@WebMedia gserg]$ less .bashrc &  
  
[1] 3070  
  
[gserg@WebMedia gserg]$ ps -efl | grep less  
  
0 T gserg 3070 3018 0 80 0 - 1004 finish 17:56 pts/3 00:00:00 less .bashrc  
  
[gserg@WebMedia gserg]$ nice -n 20 less .bashrc &  
  
[1] 3081  
  
[gserg@WebMedia gserg]$ ps -efl | grep less  
  
0 T gserg 3081 3018 0 99 19 - 1003 finish 18:01 pts/3 00:00:00 less .bashrc  
  
Сравнявайки цифрите на приоритета, отбелязваме че командата less в първия случай се е изпълнявала с приоритет 80, а във втория – 99. В такъв смисъл командата nice и свършила своята задача – понижила е приоритета на задачата. Нужно е да се отбележат само някои особености при изпълнение на nice.  Първо командата понижава приоритета колкото е възможно(например 19 вместо 20). Второ – да повишава приоритета на задачата може само root.  
  
Още една команда:  
  
nohup command – позволява на процеса да продължи изпълнението даже при загуба на управляващ терминал (SIGHUP). Тази команда е изгодно да се използва когато е необходимо да се изпълни продължително действие. Можете да я стартирате и затварята терминалния сеанс, а тя при това ще продължава да се изпълнява. Изхода от командата се съдържа във файл в текущата директория.  
  
Най често използваната команда в случая може да се каже, че е kill:  
  
kill -SIGNAL pid – изпраща сигнал до процеса с идентификатор pid. Ако сигнала не е указан, командата изпраща на процеса сигнал SIGTERM. Ето пример за използването и:  
  
[gserg@WebMedia gserg]$ less &  
  
[1] 1352  
  
[gserg@WebMedia gserg]$ ps  
  
PID TTY TIME CMD  
  
1322 pts/2 00:00:00 bash  
  
1352 pts/2 00:00:00 less  
  
1353 pts/2 00:00:00 ps  
  
[gserg@WebMedia gserg]$ kill -SIGKILL 1352  
  
[gserg@WebMedia gserg]$ ps  
  
PID TTY TIME CMD  
  
1322 pts/2 00:00:00 bash  
  
1355 pts/2 00:00:00 ps  
  
[1]+ Killed less  
  
Не толкова популярна като kill е командата killall:  
  
killall -s SIGNAL процес – изпраща сигнал до всички процеси със зададеното име. Ако не е зададен сигнал то се изпраща SIGTERM.  
  
Сигнала за тази команда, трябва да се указва без представката SIG. За получаване на отговор в цифров вид и името на сигнала се използва опцията -l на командата killall. Ето пример за използването и:  
  
[gserg@WebMedia gserg]$ less ./.bashrc&  
  
[1] 1374  
  
[gserg@WebMedia gserg]$ less ./.bashrc&  
  
[2] 1375  
  
[1]+ Stopped less ./.bashrc  
  
[gserg@WebMedia gserg]$ less ./.bashrc&  
  
[3] 1376  
  
[2]+ Stopped less ./.bashrc  
  
[gserg@WebMedia gserg]$ ps  
  
PID TTY TIME CMD  
  
1322 pts/2 00:00:00 bash  
  
1374 pts/2 00:00:00 less  
  
1375 pts/2 00:00:00 less  
  
1376 pts/2 00:00:00 less  
  
1377 pts/2 00:00:00 ps  
  
[3]+ Stopped less ./.bashrc  
  
[gserg@WebMedia gserg]$ killall -s KILL less  
  
[1] Killed less ./.bashrc  
  
[2]- Killed less ./.bashrc  
  
[3]+ Killed less ./.bashrc  
  
[gserg@WebMedia gserg]$

Използване на Pgrep, pkill, убийство и работни места

Пример 1:

1) Като основен потребител, създайте задача, изпълнявана във фона на текущия ви терминал. Изпълнете скрипта за създаването на този програмен процес.

[root @ localhost] # (докато е вярно; направете echo "Моята програма"> ~ / output.file; done) &

2) Преглед на текущите задачи, изпълнявани във фона на вашия терминал.

[root @ localhost ~] # задания

[1] - Работещи (докато е вярно; направете <echo -n "Моята програма" >> ~ / output.file; done) &

3) Спрете изпълнението на процеса, без да убивате процеса, като използвате командата kill.

[root @ localhost] # kill-SIGSTOP% 1 (% 1 е номерът на заданието, ако заданието е 2, то ще бъде% 2)

4) Преглед на спряните задачи във фонов режим.

[root @ localhost ~] # задания

[1] + Стоп (докато е вярно; направете echo -n "Моята програма" >> ~ / output.file; done) &

5) Стартирайте процеса отново с командата kill.

[root @ localhost] # kill-SIGCONT% 1

6) Убийте процеса, без да допускате блокиране на командата kill.

[root @ localhost] # kill -SIGKILL% 1

Пример 2:

1) Изтеглете и инсталирайте услугата httpd.

[root @ localhost] # yum install httpd

2) Стартирайте услугата httpd (или се уверете, че тя работи).

[root @ localhost] # systemctl стартиране на httpd

 3) Като root потребител, grep за всички процеси, които се изпълняват като root потребител и показват имената на процесите.

[root @ localhost] # pgrep -u root -l

4) Като потребител потребител стартирайте vi програмата на терминала.

[user @ localhost] # vi

5) Като потребител на root, във втория ви прозорец на терминала, grep за всички процеси, изпълнявани под потребителя "потребител" и включете имената на процесите.

[root @ localhost ~] # pgrep -u потребител -l

3690 dconf-service

3694 vim

... и т.н. допълнителна продукция отсече.

6) Като root потребител, grep за "httpd" процес.

[root @ localhost] # pgrep httpd

7) Като потребител на root, убийте всички потребителски процеси на "потребител" и стартирайте този потребител от системата.

[root @ localhost ~] # w

 14:15:59 нагоре 20:55, 4 потребители, средно натоварване: 0.00, 0.01, 0.05

ПОТРЕБИТЕЛ TTY ВХОД @ IDLE JCPU PCPU КАКВО

потребител pts / 0 Mon22 2: 58m 0.05s 0.05s bash

[root @ localhost] # pkill -t pts / 0

Това убива всеки процес, стартиран от потребителския терминал, но не стартира потребителя. Сега открийте всички работещи процеси, които трябва да бъдат Bash или SSH.

[root @ localhost] # потребител на pgrep -u

[root @ localhost] # pkill -u потребител ssh