Temat: Harmonogram zadań w Linux

cron jest opartym na czasie programem do harmonogramowania zadań w systemach operacyjnych z rodziny Unixa. Może zostać wykorzystany do uruchamiania zadań (programów, komend, skryptów) o określonych godzinach, datach albo regularnie zgodnie z określonym interwałem.

Zasada działania

cron przegląda /var/spool/cron/crontabs w poszukiwaniu plików-tabel (crontab), o nazwach zgodnych z istniejącymi kontami systemowymi. Znalezione tabele ładuje do pamięci.
cron ładuje plik konfiguracyjny /etc/crontab.

W pliku /etc/crontab wpisane są dodatkowe reguły, które uruchamiają zadania co godzinowe, codzienne, cotygodniowe i comiesięczne. Zadania te wpisywane są w postaci skryptów do katalogów: /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly, /etc/cron.monthly.

cron budzi się co minutę, sprawdzając wszystkie załadowane tabele, czy jakaś komenda w tej minucie nie powinna być wywołana. Podczas wywoływania komend, ich wyjście (oraz standardowy strumień błędów) jest przesyłane pocztą elektroniczną do właściciela tabeli.

cron co minutę sprawdza, czy czas modyfikacji katalogu /var/spool/cron/crontabs (lub czas modyfikacji /etc/crontab) był zmieniony, a jeśli tak, to cron sprawdzi czasy modyfikacji tabel i przeładuje wszystkie te, które były ostatnio zmienione. Dlatego nie jest konieczne restartowanie demona po wprowadzeniu zmian w plikach.

W plikach-tabelach zadania do wykonania opisywane są przez sześć pól oddzielonych spacją lub tabulatorem. Pierwsze pięć pól służy określeniu czasu, natomiast ostatnie pole to polecenie do wykonania. Kolejne pola czasu określają: minuty, godziny, dni, miesiące, dni tygodnia.

Zapis:

```
* * * * * polecenie
```

oznacza polecenie wykonywane co minutę.

Zapis:

```
5 4,22 */2 * 1-5 polecenie
```

oznacza polecenie wykonywane w 5 minucie 4 i 22 godziny w każdy dzień parzysty, jeśli ten dzień nie jest sobotą lub niedzielą.

crontab - tabela programu cron posiadająca specjalny format oraz nazwa programu służącego do jej edycji.

Niektóre dystrybucje Linuksa mają dodatkową tabelę systemową umieszczoną w pliku /etc/crontab.
- sposób obsługi skryptów z katalogów /etc/cron.*:

/etc/cron.hourly - skrypty wykonywane co godzinę

/etc/cron.daily - skrypty wykonywane codziennie

/etc/cron.weekly - skrypty wykonywane raz w tygodniu

/etc/cron.monthly - skrypty wykonywane raz w miesiącu

W powyższych katalogach należy umieścić plik ze skryptem do wykonania.

Przykładowa tabela crontab z systemu GNU/Linux właściwa dla danego użytkownika:

Używaj /bin/sh do wywoływania komend. Nieważne co jest w /etc/passwd.

SHELL=/bin/sh

Przesyłaj wyjście do użytkownika wojtek

MAILTO=wojtek

Użyj następującej wartości zmiennej PATH

PATH=/sbin:/bin:/usr/sbin:/usr/bin:/home/wojtek/bin

Uruchamiaj się 5 minut po północy, codziennie

5 0 * * * $HOME/bin/daily.job >> $HOME/tmp/out 2>&1

Uruchamiaj się o 14:15 pierwszego dnia każdego miesiąca -- wyjście jest przesyłane do wojtka

15 14 1 * * $HOME/bin/monthly

Denerwuj Stefana od poniedziałku do piątku o godzinie 22:00

0 22 * * 1-5 echo "Stefanie, jest już 22:00. Gdzie są Twoje dzieci?!" | mail -s "Wybiła 22:00" stefan@example.com

23 0-23/2 * * * echo "uruchamiaj 23 min po polnocy, 2, 4 ..., codziennie"

5 4 * * sun echo "Uruchamiaj się 5 po 4 w niedziele"

0 10 * * 1,3 echo "Uruchamiaj się w każdy poniedzialek i srode o 10.00"

*/2 * * * * echo "Uruchamiaj się co 2 minuty"

lub z wykorzystaniem zapisu słownego z wykorzystaniem znaku "@" tzn.:

@reboot echo "System uruchomił się!" -polecenie uruchamiane każdorazowo po uruchomieniu systemu

@yearly echo "Minął kolejny rok!" -odpowiednik zapisu "0 0 1 1 *"

@annually echo "Minął kolejny rok!" - j.w.

@monthly echo "Minął kolejny miesiąc!" -odpowiednik zapisu "0 0 1 * *"

@weekly echo "Minął kolejny tydzień!" -odpowiednik zapisu "0 0 * * 1"

@daily echo "Minął kolejny dzień!" -odpowiednik zapisu "0 0 * * *"

@midnight echo "Minął kolejny dzień!"- j.w.

@hourly echo "Minęła kolejna godzina!" -odpowiednik zapisu "0 * * * *"

Pierwsza część pliku to definicje trzech zmiennych środowiskowych SHELL, MAILTO i PATH.
Wartość zmiennej SHELL ustala powłokę, w której cron będzie uruchamiał polecenia.
Wartość zmiennej MAILTO określa użytkownika, któremu pocztą elektroniczną wysłany zostanie raport zawierający standardowy strumień wyjścia oraz standardowy strumień błędów, jeśli wykonywane polecenie umieści coś na jednym z nich.
Zmienna PATH zawiera oddzielone dwukropkami ścieżki, których powłoka używa kolejno do odnalezienia plików wykonywalnych.

Druga część to już właściwa tabela zawierająca informacje o czasie uruchomienia i poszczególnych komendach dla każdego z zadań. W tej części linie nie będące komentarzem zawierają sześć kolumn. Pierwsze pięć określa czas uruchomienia zadania, szósta definiuje komendę, która ma zostać wykonana:

* * * * * komenda do wykonania

1 * minuta (0 - 59)

2 * godzina (0 - 23)

3 * dzień miesiąca (1 - 31)

4 * miesiąc (1 - 12)

5 * dzień tygodnia (0 - 7) (niedziela=0, poniedziałek=1, wtorek=2, ..., niedziela=7) (niedziela może być przedstawiona jako 0 lub 7)

Uwagi:

W większości implementacji niedziela może być oznaczona jako 0 lub 7.

Aby uniknąć problemów z uruchomieniem poleceń systemowych, należy w crontabie podawać pełne ścieżki do nich lub ustawić odpowiednią wartość zmiennej PATH.

Dzień wykonania komendy można wyspecyfikować na dwa sposoby: podając dzień miesiąca lub dzień tygodnia. Jeśli oba pola są ustawione, to komenda wykona się zarówno w ustawiony dzień miesiąca, jak i w ustawiony dzień tygodnia.

Wartości liczbowe możemy zapisywać w różnych formatach:

1-3 - czyli wartości 1,2,3

0-10/2 - czyli wartość 0,2,4,6,8 i 10 (co druga wartość ze zbioru od 0 do 10)

1,2,5 - czyli wartości kolejno 1,2,5

*/2 - co 2 dozwolona wartość (np. w pierwszej kolumnie będzie to 0,2,4,6...56,58)

1-3,5,6 - czyli 1,2,3 oraz 5 i 6

Tabela systemowa zawarta w pliku /etc/crontab posiada nieznacznie inną składnię: pierwsze pięć pól określa czas uruchomienia zadania, szóste pole określa nazwę użytkownika, z którego uprawnieniami zadanie zostanie uruchomione, siódme pole definiuje komendę, która zostanie wykonana.

Jako specjalny znak traktowany jest "%" (znak procent), który oznacza nową linię. Aby wyłączyć tą funkcjonalność należy wstawić przed niego znak "\".

Z powodu zmiany czasu należy unikać ustawiania zadań, które miałyby się wykonać między 2:00 a 3:00 w niedzielę. Takie ustawienie powoduje, że przy automatycznej zmianie czasu na letni zadanie się nie wykona, natomiast przy zmianie czasu z letniego na zimowy wykona się dwa razy (między 2:00 i 3:00 oraz 2a:00 i 3a:00)

at - komenda systemów z rodziny Uniksa używana do ustawienia wykonania jakiegoś polecenia o zadanej godzinie w przyszłości.

at pobiera ze standardowego wejścia listę poleceń i grupuje je w pojedyncze zadanie (ang. at-job), które zostaje wykonane o zadanym czasie. Po wykonaniu żądanej sekwencji poleceń at może wysłać poprzez e-mail powiadomienie, do użytkownika który operację zlecił. Listę poleceń do wykonania można wczytać z pliku zamiast standardowego wejścia.

at używa daemona atd, który cyklicznie sprawdza listę zadań, wykonując je o oznaczonym czasie. Demon ten może być tak skonfigurowany, aby wykonywał zaplanowane zadania, tylko gdy obciążenie systemu (mierzone jako load average) jest nie wyższe niż zadany próg.

Aby dodać nowe polecenie wystarczy za pomocą polecenia echo przekazać komendę dla at podając dodatkowo, kiedy lub za ile czasu ma się wykonać.

W poniższym przykładzie zostało zlecone uruchomienie skryptu foo.sh za godzinę.

$ echo 'foo.sh' | at now + 1 hour

Ten sam efekt uzyskamy w ten sposób. Może okazać się on pomocny, gdy mamy do wykonania parę poleceń.

$ at now + 1 hour

at> foo.sh

at> foo2.sh

at> foo3.sh

at> ^D #(Control-D zakończy wprowadzanie komend)

Inne przykłady użycia:

$ echo 'foo.sh' | at 2017-04-30 # 30 kwietnia 2017 roku

$ echo 'foo.sh' | at now + 5 minute # za 5 minut

$ echo 'foo.sh' | at now + 1 day # jutro o tej samej porze

$ echo 'foo.sh' | at 2210 # o 22:10, jeśli dziś ta godzina minęła to wykona się jutro

Polecenie atq służy do wypisania listy aktualnie zleconych zadań. W pierwszej kolumnie znajduje się identyfikator zadania, potem kolejno data zaplanowanego uruchomienia oraz osoba zlecająca.

W tym wypadku user.

$ atq

44 Fri Apr 28 20:14:00 2017 a user

45 Fri Apr 28 20:19:00 2017 a user

Gdy znane nam jest id zadania to poleceniem atrm <identyfikator_zadania> mamy możliwość usunięcia tych, które nas interesują.

$ atq

44 Fri Apr 28 20:14:00 2017 a user

45 Fri Apr 28 20:19:00 2017 a user

$ atrm 44 #usunięcie zadania o id 44

$ atq

45 Fri Apr 28 20:19:00 2017 a user

$ atrm 45 #usunięcie zadania o id 45

$ atq

$

Źródło: wikipedia