Demony cron i at

Dokumentacja do prezentacji

Źródła:

cron(8) man page
crontab(1) and crontab(5) man pages
at(1) man page
atd(8) man page
“UNIX: Administracja systemu” Eleen Frish, Wydawnictwo RM 1997, [str. 391-399]
“The Linux A-Z” Phil Cornes, Prentice Hall Europe 1997, [str. 154-156]

Autorzy:

Jarosław Matuszak, 340919
Piotr Malinowski, sXXXXXX

Spis treści:

1. Demon *cron*

1.1 Czym jest cron i do czego służy?

Cron jest usługą stworzoną dla systemów UNIX-owych. Pozwala on szeregować zadania w celu ich cyklicznego uruchamiania. Przez zadania rozumiemy wszelkiego rodzaju skrypty i polecenia. Czas uruchomienia zadania może być bardzo specyficznie określony - co do minuty. Dane wyjściowe, poleceń wywoływanych przez *cron*'a, można zapisywać do wskazanego pliku lub wysyłać na adres e-mail wybranych użytkowników. *Cron* jest instalowany domyślnie na większości systemów typu UNIX. Demon *cron* jest zazwyczaj uruchamiany wraz z systemem, ale może zostać uruchomiony ręcznie poleceniem:

```
> /etc/init.d/cron start
```
lub
```
> /etc/rc.d/init.d/cron start
```

Usługa *cron* jest obsługiwana za pośrednictwem demona, czyli procesu działającego "w tle", o nazwie *cron*. Lista poleceń i skryptów, które mają być uruchamiane przez *cron*'a znajduje się w specjalnym pliku *crontab*. Każdy użytkownik ma własny plik *crontab*, z wpisami które sam utworzył. Plik nosi taką samą nazwę jak nazwa użytkownika (z `/etc/passwd`). Na przykład dla użytkownika "root", plik ten znajdował by się tutaj:

```
/var/spool/cron/crontabs/root
/usr/spool/cron/crontabs/root #dla HP-UX
/usr/spool/cron/tabs/root #dla Linux
```

Demon *cron* przeszukuje również foldery `/etc/crontab` oraz `/etc/cron.d` (zadania systemowe użytkowników) w poszukiwaniu plików *crontabs*. Formatowanie w tych plikach jest jednak inne niż w folderach wymienionych powyżej. Niniejsza dokumentacja skupi się na formatowaniu plików z `/var/spool/cron/crontabs`, ponieważ właśnie taki sposób planowania zadań jest zalecany.

W większości wypadków demon *cron* odczytuje pliki *crontabs*, we wcześniej wymienionych folderach, tylko w momencie jego uruchamiania lub dokonywania w nich zmian. Starsze wersje *cron*'a mogą jednak odczytywać te pliki co pięć minut.

1.2 Budowa polecenia *cron*

Omówienie, w trochę większym szczególe, polecenia *cron* v4.1. Od tego momentu, jeżeli nie napisano inaczej, dokumentacja opisywać będzie obsługę *cron*'a za pomocą pakietu *ISC Cron (kiedyś nazywany Vixie Cron)*.

1.2.1 Zarys

`cron [-f] [-l] [-L loglevel]`

1.2.2 Opcje

`-f` Wymusza na usłudze aby pracowała na pierwszym planie. Domyślnie pracuje jako demon “w tle”.

`-l` Umożliwia wykorzystywanie nazw zgodnych ze standardem LSB dla plików *crontabs* w folderze `/etc/cron.d`. Parsowanie plików `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly` i `/etc/cron.monthly` nie ulega jednak zmianie. Pliki te wykorzystywane są tylko w dystrubicjach DEBIAN, dla zadań zdefiniowanych w `/etc/crontab`. Zadania w każdym z tych czterech plików wykonywane są odpowiednio co godzinę, co dzień, co tydzień i co miesiąc. Nie będą one omawiane w niniejszej dokumentacji.

`-L loglevel` Określa jakie informacje o wykonywanych poleceniach zapisywać w logach *loglevel* to suma poniższych wartości:

`(1)` logowanie rozpoczęcia każdego zadania *cron*
`(2)` logowanie zakończenia każdego zadania *cron*
`(3)` logowanie każdego zadania *cron* zakończonego porażką (kod wyjścia != 0)
`(4)` logowanie numeru procesu każdego zadania *cron*

Domyślnie *loglevel* = 1. Aby zupełnie wyłączyć zapisywanie logów ustaw *loglevel* = 0 (wszelkie błędy zadań cron będą logowane nie zależnie od wartości *loglevel*). Jeżeli *loglevel* = 15, to logowane będą wszystkie zdarzenia. Logowanie zostanie omówione w większym szczególe w dalszej częśći dokumentacji.

1.2.3 Uwagi

Jeżeli zegar zostanie przesunięty o mniej niż 3 godziny, to *cron* zachowa się w następujący sposób:

- Jeżeli przesuniemy czas o < 3 godzinydo przodu, to wszystkie zadanie, które miały zostać wykonane w “pominiętym” czasie, zostaną wykonane tuż po tej przerwie.
- Jeżeli przesuniemu czas o < 3 godziny do tyłu, to zadania, których wykonanie wypada na czas “powtórzony” nie zostaną wykonane dwukrotnie.

Cron będzie korzystał z `/etc/timezone`, w celu określenia strefy czasowej, o ile taki plik istnieje.

Dodatkowe uwagi oraz listę istotnych zmian dla dystrubucji DEBIAN można znaleźć w *man cron(8)*.

1.2.4 Konfiguracja

Plik konfiguracyjny *cron* znajduje się w /etc/default/cron. Wygląda on tak:

```
--------------------------/etc/default/cron--------------------------
# Cron configuration options

# Whether to read the system's default environment files (if present)
# If set to "yes", cron will set a proper mail charset from the
# locale information. If set to something other than 'yes', the default
# charset 'C' (canonical name: ANSI_X3.4-1968) will be used.
#
# This has no effect on tasks running under cron; their environment can
# only be changed via PAM or from within the crontab; see crontab(5).

READ_ENV="yes"

# Extra options for cron, see cron(8)
#
# For example, to enable LSB name support in /etc/cron.d/, use
# EXTRA_OPTS='-l'
#
# Or, to log standard messages, plus jobs with exit status != 0:
# EXTRA_OPTS='-L 5'
#
# For quick reference, the currently available log levels are:
#   0   no logging (errors are logged regardless)
#   1   log start of jobs
#   2   log end of jobs
#   4   log jobs with exit status != 0
#   8   log the process identifier of child process (in all logs)
#
#EXTRA_OPTS=""
-----------------------------------------------------------------------
```

Jeżeli chcemy aby *cron* mógł rejestrować/logować swoje działania, to w pliku konfiguracyjnym musimy dodać linię:

```
CRONLOG=YES
```

1.3 Planowanie zadań - *crontab*

Użytkownik nie edytuje plików zawierających zaplanowane zadania ręcznie. Zamiast tego powinien on używać polecenia *crontab*:

1.3.1 Zarys

`crontab [-u user] file`
`crontab [-u user] [-l | -r | -e] [-i]`

1.3.2 Opcje

`-u user` Określa, któremu użytkownikowi (*user*) zainstalować plik z zaplanowanym zadaniami. Dzięki temu *root* może wybrać użytkownika, w imieniu którego *cror* będzie wykonywał zadania. Odpowiedni plik *crontab* znajdzie się w katalogu *crontabs* wybranego użytkownika.

`-l` Wyświetla wszystkie zadania znajdujące się aktualnie w pliku *crontab*.

`-r` Usuwa wszystkie zadania z aktualnego pliku *crontab*.

`-e` Wykorzystywane do "ręcznego" dodawania/zmieniania/usuwana zadań z aktualnego pliku crontab. Po zatwierdzeniu polecenia z tym parametrem, otworzy się domyślny edytor tekstu (ze zmiennej środowiskowej `VISUAL` lub `EDITOR`), w którym będziesz mógł dokonać pożądanych zmian. Po opuszczeniu edytora, plik *crontab* z wprowadzonymi zmianami, zostanie automatycznie zainstalowany.

`-i` Wykorzystywane wraz z opcją *-r*. Powoduje wygenerowanie zapytania tak/nie przed usunięciem pliku *crontab*.

1.3.3 Uwagi

Wykaz najważniejszych użytkowników systemowych i rodzaj zadań, które powinny być uruchamiane w ich imieniu przez *cron*'a (z opcją *-u*):

`root` Zadania związane z ogólnym funkcjonowaniem systemu, czyszczenie systemu plików, niektóre czynności związane z rozliczaniem.

`adm` Rozliczanie oraz monitorowanie wydajności systemu.

`uucp` Zadania związane z Unix-to-Unix Copy.

`sys` Śledzenie wydajności (IRIX) oraz wykonywanie lokalnych zadań.

1.3.4 Formatowanie plików *crontab*

Każda pozycja (linia) w pliku *crontab*, mówi tyle co "wykonaj takie polecenie, o takiej godzinie, w taki dzień". Zadaniem demona *cron*, jest wykonanie tych poleceń w określonym czasie. Każda pozycja w pliku *crontab* ma następującą postać:

```
  1        2             3             4            5             6
minuta  godzina  dzień_miesiąca  miesiąc  dzień_tygodnia  polecenie
```

Poszczególne pola oddzielone są spacjami. Jednakże ostatnie pole, *polecenie*, może zawierać spacje (oznacza to, że pole *polecenie* składa się ze wszystkiego co następuje za spacją po polu

dzień_tygodnia). Pozostałe pola nie mogą zawierać spacji.

Znaczenie pierwszych 5 pól:

Pole	**Znaczenie**	**Zakres**
`1`	Minuty po pełnej godzinie	`0-59`
`2`	Godzina	`0-23 (0 = północ)`
`3`	Dzień miesiąca w postaci numerycznej	`1-31`
`4`	Miesiąc	`1-12`
`5`	Dzień tygodnia	`0-7 (0 lub 7 to niedziela)`

Każda pozycja w powyższych polach może być pojedyńczą liczbą, parą liczb połączonych myślnikiem (taki zapis oznacza zakres), listą liczb lub zakresów oddzielonych przecinkami lub gwiazdką (`*`), która oznacza wszystkie możliwe wartości pola.

Jeżeli pierwszym znakiem linii jest `#`, to cała pozycja jest ignorowana przez *cron*'a. Zwróć uwagę na to, że komentarz z użyciem `#` nie może znajdować się w tej samej linii co pozycja pliku *crontab* (komentarz będzie traktowany jako część polecenia).

Pole *polecenie* powinno zawierać dowolne polecenie UNIX'a lub grupę poleceń (oddzielonych średnikami). Każda pozycja pliku *crontab* może być dowolnie długa, ale musi znajdować się w jednej linii. Jeżeli w poleceniu znajduje się znak procent (`%`), *cron* będzie traktował dowolny następujący po tym znaku tekst jako standardowe wejście polecenia. Dodatkowy znak procentu może służyć do rozdzielania tekstu na poszczególne linie, np.

```
%Hello...%world!
```

będzie interpretowany jako:

```
Hello...
world!
```

Jeżeli interpretator polecenia nie jest wskazany, to używany jest interpretator domyślny (zazwyczaj Bourne'a). Jeżeli chcemy wskazać interpretator możemy jako polecenie wpisać:

```
/usr/bin/perl my_perl_script
```

W ten sposób mówimy *cron*'owi, że ma do czynienia ze skryptem języka *perl*.

1.3.5 Pakiet *ISC Cron*

ISC Cron jest pakietem obsługującym standardowe cechy usługi *cron* oraz rozszerzającym format poszczególnych pozycji pliku *crontab*. Przykładowe właściwości tego pakietu:

- Miesiące i dni mogą być podawane za pomocą ich nazw, skróconych do pierwszych trzech liter, np.: `mon`, `sun`, `jan`, `may`, `feb`.
- Wartości kroku mogą być podawane za pomocą przyrostka `/n`, występującego po liczbie. Stąd napis "`8-18/2`" oznacza "co dwie godziny, od 8 do 18", a "`*/5`" w polu minut oznacza "co pięć minut".
- W pliku *crontab* mogą być definiowane zmienne środowiskowe z wykorzystaniem normalnej składni interpretatora Bourne'a. Zmienna o nazwie `MAILTO`, może być użyta do określania, kto będzie otrzymywał pocztę rozsyłaną przez *cron*'a. Jeżeli wpiszemy "`MAILTO=root`", to wiadomości będzie otrzymywał root. Wpisanie samego "`MAILTO=`" zawiesza wysyłanie poczty przez *cron*'a. Używanie znaku komentarza (`#`) jest zabronione w tej samej linii co definicja zmiennej środowiskowej.

  Domyślnie:

  `SHELL = /bin/sh`
  `LOGNAME` i `HOME` pobierane są z `/etc/passwd` (LOGNAME nie może być nadpisany przez użytkownika)

Pliki z definicjami pozycji crontab są przechowywane w `/var/spool/cron/tabs` (niech cię nie zmylą pliki w `/var/spool/cron/crontabs`).

1.4 Pliki rejestrujące dla *cron*'a

Większość wersji *cron*'a posiada mechanizm utrwalania wszytskich wykonywanych przez niego działań, zwykle wykorzystywane są do tego logi (pliki rejestrujące).

Poniżej prezentujemy wykaz właściwości i strategie rejestrowania działań podejmowanych przez *cron*'a w poszczególnych systemach oraz lokalizację plików rejestrujących:

`AIX`	Automatycznie w pliku `/var/adm/cron/log`
`SCO UNIX`	Opcjonalnie w pliku `/usr/lib/cron/log`
`Digital UNIX`	Automatycznie w pliku `/var/adm/cron/log`
`IRIX`	Automatycznie w pliku `/var/cron/log`
`HP-UX`	Automatycznie w pliku `/usr/lib/cron/log`
`Solaris`	Opcjonalnie w pliku `/var/cron/log`

`SunOS`	Opcjonalnie za pośrednictwem programu `syslog`
`Linux`	Automatycznie w pliku `/var/spool/cron/log`

Jeżeli logowanie akcji *cron*'a jest włączone, zalecane jest regularne zmniejszanie zawartości tych plików.

1.5 Bezpieczeństwo

Podstawowymi problemami z jakimi boryka się usługa *cron* w kwestii bezpieczeństwa są:

- Odpowiednie zabezpieczenie plików *crontab*:
  ►Rozwiązaniem może być blokowanie praw pisania do tych plików.
- Upewnienie się, że żaden nie pożądany użytkownik nie uruchamia groźnych poleceń za pomocą *cron*'a:
  ►Rozwiązaniem tego problemu jest odpowiednia administracja plikami *cron.allow* i *cron.deny*, opisanych w następnym śródtytule.

1.6 Ustalanie praw dostępu do usługi *cron*

Możliwość korzystania z usługi *cron* można konfigurować za pomocą plików *cron.allow* i *cron.deny*. Oba pliki zawierają po jednej nazwie użytkownika w każdej linii. Pliki te działają w następujący sposób:

- Jeżeli istnieje plik *cron.allow*, to użytkownik musi być wymieniony na liście, aby mógł korzystać z polecenia *crontab*.
- Jeżeli *cron.allow* nie istnieje, a istnieje *cron.deny*, to z *crontab* mogą korzystać wszyscy użytkownicy nie wymienieni w *cron.deny*. Jeżeli *cron.deny* będzie pusty, to wszyscy mogą korzystać z *cron*'a.
- Jeżeli oba pliki nie istnieją, to tylko *root* ma dostęp do *cron*'a.

Pliki te blokują możliwość wywołania polecenia *crontab*. Oznacza to, że wszystkie nie usunięte zadania będą nadal wykonywane, nawet jeżeli użytkownik nie ma dostępu do *cron*'a.

Lokalizacja plików w systemach UNIX:

`HP-UX, SCO UNIX`	`/usr/lib/cron/cron.{allow,deny}`
`IRIX,Solaris`	`/etc/cron.d/cron.{allow,deny}`
`SunOS`	`/var/spool/cron/cron.{allow,deny}`
`Linux`	`/var/spool/cron/{allow,deny}`
`AIX, Digital UNIX`	`/var/adm/cron/cron.{allow,deny}`

1.7 Przykłady zastosowań *cron*'a

1.7.1 Zaplanowanie pierwszego zadania

Na potrzeby tego przykładu stwórzmy następujące katalogi i pliki pomocnicze:

```
> cd $HOME
> mkdir {logs,work}
> touch ./work/file{1,2,3,4}
```

W katalogu `logs` będziemy przechowywać logi generowane przez nasz skrypt. Każdy plik logu będzie nosił nazwę:

```
data_godzina
```

W pliku tym będzie znajdować się lista wszystkich plików, jakie znajdowały się w folderze *work* o danej porze, określonej przez nazwę logu. Na potrzeby tego przykładu, będziemy generować taki log co minutę.

Stwórzmy zatem prosty skrypt:

```
> vi snapshot

----------------------------/$HOME/snapshot-------------------------
#! /bin/sh
DATE=`eval date +%e.%m.%Y_%H:%M:%S`
touch $HOME/logs/$DATE
ls $HOME >> $HOME/logs/$DATE
-------------------------------------------------------------------
```

Musimy jeszcze nadać odpowiednie prawa dla naszego skryptu:

```
> chmod 700 snapshot
```

Stwórzmy teraz plik *mytab*, który będzie naszym plikiem *crontab*:

```
> vi mytab

----------------------------/$HOME/mytab----------------------------
* * * * * $HOME/snapshot
#* * * * * /bin/sh $HOME/snapshot
-------------------------------------------------------------------
```

Pierwsza linia mówi tyle co: wykonuj co minutę, każdego dnia skrypt *snapshot*. Druga, zakomentowana linia, robi dokładnie to samo, ale wskazaliśmy wprost interpretator dla tego polecenia. Zróbmy teraz z tego pliku nasz *crontab*:

```
> crontab mytab
```

Od tej chwili co minutę tworzony będzie nowy log. Aby mieć pewność, że zadanie rzeczywiście znajduje się w pliku *crontab* wywołaj:

```
> crontab -l
```

Aby usunąć wszystkie zadania wykonaj:

```
> crontab -r -i
```

Jeżeli nie chcesz usunąć wszystkich wpisów, a np. usunąć pierwszą linię i odkomentować drugą, to wywołaj:

```
> crontab -e
```

Uruchomi się teraz edytor tekstu, w którym możesz wprowadzić dowolne zmiany (usuwanie/zmienianie/komentowanie linii). Po zapisaniu zmian i wyjściu z edytora, zostaną one automatycznie zapisane w pliku *crontab*.

Załóżmy, że poza własnymi logami, chcemy odnotowywać każde wywołanie naszego skryptu przez *cron*'a. Zacznijmy od włączenia logowania zdarzeń, poprzez modyfikację pliku konfiguracyjnego *cron*'a:

```
> vi /etc/default/cron
```

Dodajmy tam linię:

```
CRONLOG=YES
```

Teraz musimy uruchomić *cron*'a ponownie z odpowiednią wartością parametru *loglevel*:

```
> /etc/init.d/cron stop
> cron -L 1
```

Nasze logi są domyślnie zapisywane w */var/log/syslog*. Aby odnaleźć wpisy dotyczące *cron*'a wykonaj:

```
> grep CRON /var/log/syslog
```

1.7.2 Pozostałe przykłady

Poniżej znajduje się klika przykadów pozycji *crontab* wraz z krótki opisem co każda z nich robi:

```
1  0,15,30,45 * * * * (echo `\c`; date; echo "") > /dev/console
2  0,10,20,30,40,50 8-18/2 * * * /usr/lib/atrun
3  0 0 * * * find / -name "*.bak" -type f -atime +7 -exec rm {} \;
4  0 4 * * * /bin/sh /var/adm/mon_disk 2>&1 /var/adm/disk.log
5  0 2 * * * /bin/sh /usr/local/sbin/sec_check 2>&1 | mail root
6  30 11 31 12 * /etc/wall%Happy new year!%Let it be great!
7  @monthly /bin/sh /var/adm/monthly 2>&1 | mail root
```

Wyjaśnienie:

1 Wyświetla na konsoli co piętnaście minut aktualną datę i godzinę. Kilka poleceń jes ujętych w nawiasie w celu skierowania ich wspólnego wyjścia na konsole.

2 Uruchamia *atrun* codziennie, co dwie godziny od 8 do 18.

3 Uruchamia polecenia *find* w celu odnalezienie i usunięcia wszystkich plików *.bak, z których nie korzystano od siedmiu dni.

4,5 Wykonuje skrypty odpowiednio o 4:00 i 2:00 nad ranem. W tym przypadku wskazan wprost interpretator jaki ma być wykorzystany.

6 Uruchamia w nowy rok skrypt, który otrzymuje na wejściu tekst "`Happy new year!\nLet it be great!`".

7 Uruchamia raz na miesiąc skrypt *monthly* i wysyła maila do *root*'a.

Jak pokazano w ostatni przykładzie, zamiast pierwszych pięciu pól możemy wykorzystać specjalne łańcuchy:

Łańcuch	**Znaczenie**
`@reboot`	Wykonaj raz, po uruchomieniu systemu.
`@yearly` lub `@annually`	Wykonaj raz na rok (pierwszego stycznia, zaraz po północy)
`@monthly`	Wykonaj raz na miesiąc (pierwszego dnia miesiąca)
`@weekly`	Wykonaj raz na tydzień (w niedzielę)

`@daily` lub `@midnight`	Wykonaj raz na dzień (po północy)
`@hourly`	Wykonaj raz na godzinę

2. Demon *at*

2.1 Czym jest *at* i do czego służy?

At jest komendą systemów operacyjnych z rodziny Unix/Linux. Czyta polecenia ze standardowego wejścia lub podanego pliku, które mają zostać wykonane w terminie późniejszym przy użyciu `/bin/sh`. W przeciwieństwie do *cron* nie pozwala na cykliczne wykonywanie zadań - wykonuje dane zadanie jednorazowo w zadanym terminie.

Oprócz *at*, omówimy również komendy ściśle spokrewnione: *batch, atq, atrm.*

2.1.1 Zarys

```
at [-V] [-q kolejka] [-f plik] [-mldbv] CZAS
at -c zadanie [zadanie...]
atq [-V] [-q kolejka]
atrm [-V] zadanie [zadanie...]
batch [-V] [-q kolejka] [-f plik] [-mv] [CZAS]
```

2.1.2 Opis

`at` Wykonuje polecenia o zadanym czasie.

`atq` Pokazuje oczekujące zadania użytkownika, chyba że jest on superużytkownikiem w tym przypadku wyświetlane są zadania wszystkich użytkowników. Postać wierszy wynikowych (po jednym na każde zadanie): numer zadania, data, godzina klasa zadania.

`atrm` Usuwa zadania o zadanych identyfikatorach (numerach).

`batch` Wykonuje polecenia, gdy pozwala na to poziom obciążenia systemu; inaczej mówiąc, gdy średnie obciążenie spada poniżej 0.8 lub wartości określonej w wywołaniu *atrun*.

At pozwala na dość złożone określanie czasu, wychodząc poza standard POSIX.2 Akceptuje czasy w postaci HH:MM nakazujące wykonanie zadania o zadanej godzinie. (Jeśli czas ten już minął, to przyjmowany jest następny dzień.) Możesz też podać *midnight* [północ], *noon* [południe] lub *teatime* ["czas herbatki": czwarta po południu]. Możesz używać przyrostków [tłum: decydują one równocześnie o użyciu zegara 12-godzinnego zamiast 24-godzinnego] AM lub PM do zapisu uruchomień porannych lub wieczornych. Można też wskazać, w jakim dniu ma zostać uruchomione zadanie, podając datę w postaci nazwa-miesiąca dzień z opcjonalnym rokiem, lub też w formacie MMDDYY, MM/DD/YY czy DD.MM.YY. Określenie daty musi występować po określeniu godziny (pory dnia). Możesz również podawać czasy takie jak *now* [teraz] + liczba jednostek-czasu, gdzie jednostkami-czasu mogą być *minutes* [minuty], *hours* [godziny], *days* [dni], lub *weeks* [tygodnie]. Można też nakazać at uruchomienie zadania w dniu dzisiejszym kończąc określenie czasu słowem *today* [dzisiaj], czy jutrzejszym, używając przyrostka

tomorrow [jutro].

Na przykład, by uruchomić zadanie za trzy dni, o godzinie 4 po południu, powinieneś wykonać:

```
> at 4pm + 3 days
```

uruchomienie zadania o 10:00, 31 lipca:

```
> at 10am Jul 31
```

jutro o pierwszej w nocy:

```
> at 1am tomorrow
```

Dokładną definicję specyfikacji czasu można znaleźć w pliku `/usr/doc/packages/at/timespec`.

Zarówno *at* jak i *batch*, czytają i wykonują polecenia ze standardowego wejścia lub z pliku określonego opcją *-f*. Z czasu wywołania poleceń zachowywane są katalog roboczy, środowisko systemowe (z wyjątkiem zmiennych `TERM`, `DISPLAY` i `_`) oraz domyślne prawa dostępu (*umask*). Polecenie *at* - czy *batch* - wywołane z powłoki (1) zachowają bieżący identyfikator użytkownika. Wyniki kierowane przez zadane polecenia na standardowe wyjście lub wyjście raportowania błędów zostaną skierowane do skrzynki pocztowej użytkownika (przekazane pocztą elektroniczną). Przesyłki te zostaną dostarczone przy pomocy polecenia `/usr/sbin/sendmail`. Jeżeli *at* wykonywane jest z powłoki (1), to korespondencję otrzyma właściciel powłoki zgłoszeniowej (*login shell*).

Superużytkownik może zawsze posługiwać się opisywanymi poleceniami. Dla pozostałych użytkowników, zezwolenia na użycie at określane są przez pliki `/etc/at.allow` i `/etc/at.deny`.

Jeżeli istnieje plik `/etc/at.allow`, to wyłącznie użytkownicy, których nazwy doń wpisano, mają prawo uruchamiania *at*.

Jeśli nie istnieje `/etc/at.allow`, to sprawdzany jest plik `/etc/at.deny`, a każdemu użytkownikowi, którego w nim nie wymieniono, zezwala się na używanie *at*. [tłum: tzn. użytkownicy ujęci w `/etc/at.deny` mają zakaz uruchamiania opisywanych poleceń.]

Jeśli nie istnieje żaden z plików kontrolujących dostęp do polecenia *at*, to posługiwać się nim może wyłącznie superużytkownik.

Pusty plik `/etc/at.deny` oznacza, że pozwala się na używanie opisywanych poleceń wszystkim użytkownikom. Jest to konfiguracja domyślna.

2.1.3 Opcje

`-V` Wyświetla numer wersji programu na standardowe wyjście błędów.

`-q` Kolejka;
Używa zadanej kolejki. Określenie kolejki składa się z pojedynczej litery; dopuszczaln są określenia kolejki w zakresie od a do z oraz od A do Z. Kolejka a jest kolejk domyślną dla at, zaś kolejka b domyślną dla batch. Kolejki opisywane kolejnymi literan uruchamiane są z wzrastającym priorytetem. Specjalna kolejka "=" zarezerwowana jest dla zadań obecnie wykonywanych.
Jeśli zadanie wysyłane jest do kolejki określonej dużą literą, to traktowane jest jakby by wysłane do wykonania o tym czasie przez polecenie *batch*. Jeśli użyto konkretnej kolejk w poleceniu *atq*, to pokaże ono tylko zadania oczekujące w tej kolejce.

`-m` Po zakończeniu zadania wyślij pocztą powiadomienie do użytkownika, nawet jeś zadanie nie wysłało danych na wyjście.

`-f` Plik;
Czyta zadania z pliku a nie ze standardowego wejścia.

`-l` Jest skrótem (aliasem) dla *atq*.

`-d` Jest skrótem dla *atrm*.

`-v` Pokazuje czas, o jakim zostanie wykonane zadanie. Czasy wyświetlone zostaną w formacie "`1997-02-20 14:50`", chyba że ustawiono zmienną środowiska `POSIXLY_CORRECT`; wówczas będzie to "`Thu Feb 20 14:50:00 1996`".

`-c` Wysyła [jak polecenie *cat (1)*] podane w wierszu poleceń zadania na standardowe wyjście.

2.2 Pliki

/var/spool/atjobs
/var/spool/atspool
/proc/loadavg
/var/run/utmp
/etc/at.allow
/etc/at.deny

Zobacz także:

cron (1), nice (1), sh (1), umask (2), atd (8).

2.3 Błędy

Poprawne działanie *batch* w Linuksie zależy od obecności typu katalogu `proc`- montowanego w */proc*.

Jeżeli plik `/var/run/utmp` nie jest dostępny lub jest uszkodzony, albo jeśli podczas wywoływania *at* użytkownik nie jest zalogowany, to poczta wysyłana jest do użytkownika o identyfikatorze znalezionym w zmiennej środowiska `LOGNAME`. Jeżeli nie jest ona zdefiniowana lub jest pusta, to przyjmowany jest bieżący identyfikator użytkownika.

At i *batch* w obecnej implementacji są nieużyteczne w sytuacji, kiedy użytkownicy rywalizują o zasoby. Jeśli jest tak w przypadku Twojego systemu, powinieneś rozważyć inne rozwiązanie systemu wsadowego, takie jak *nqs*.