Zielonogórska Grupa Użytkowników Linuksa
http://zgul.ime.uz.zgora.pl

Raz na zawsze

Zapomnij o wszystkim, o czym musisz pamiętać

Tomasz Jakub Skrynnyk
T.Skrynnyk@zgul.ime.uz.zgora.pl
http://zgul.ime.uz.zgora.pl/~tskrynnyk/

Spis treści

Wersja: 1.12
9 sierpnia 2004 roku

1. Co ja tu miałem napisać?

Moment... Chwila... Uff! Przypomniałem sobie. Na początku chcę napisać to, co powinno znaleźć się w standardowej przedmowie, wprowadzeniu i chyba w streszczeniu, którego tu nie ma (niestety, nie pamiętam dlaczego).

1.1. Wprowadzenie

Ten artykuł powstał w związku z prowadzeniem przeze mnie wykładu w ramach spotkań organizowanych przez Zielonogórską Grupę Użytkowników Linuksa. Spora część dokumentu powstała już w trakcie przygotowań, reszta niedługo po nim. Sam wykład odbył się 7 lipca 2004 r. w Zielonej Górze.

1.1.1. O czym jest ten artykuł

Tematem artykułu jest wykorzystanie Linuksa do przypominania. Pokazane są tu sposoby wykorzystania istniejących narzędzi oraz dostępnego oprogramowania. Zademonstrowane zostały też możliwości tworzenia własnych rozwiązań. Skupiam się tu tylko i wyłącznie na praktycznym wykorzystaniu odpowiednich narzędzi świadomie pomijając choćby dokładne opisy stosowanych programów (sposób działania, dostępne opcje, konfiguracja itp.). Pamiętaj, proszę, o tym w trakcie lektury. Rolą tego dokumentu jest tylko wprowadzenie w świat terminarzy i tylko takich, które nie mają zbyt wielu wymagań, pozwalają na pełną automatyzację (dają się łączyć z innymi narzędziami, pracują jako demony) i – przede wszystkim – są mi dobrze znane.

1.1.2. I komu się przyda

Artykuł jest adresowany do wszystkich użytkowników Linuksa, którzy obsługę systemu opanowali co najmniej w stopniu średnim i chcą zapomnieć o wszystkim, o czym muszą pamiętać.

1.2. Prawa autorskie i licencje

[1] Dokładne znaczenie terminu *kopia jawna* wyjaśnia definicja zawarta w licencji GNU Wolnej Dokumentacji.

1.3. Kontakt z autorem

Jeżeli chcesz się ze mną skontaktować w sprawie artykułu, wykładu, albo tak sobie, możesz to zrobić pisząc na adres T.Skrynnyk@zgul.ime.uz.zgora.pl. Uprzedzam, że bardzo szanuję czas swój i innych, więc kontakt z powodu *tak sobie* najprawdopodobniej będzie przeze mnie zignorowany.

2. Idea – coś z głowy, czyli z niczego

2.1. Syndrom słonia Trąbalskiego

Już bardzo dawno temu zauważyłem, że ten – skądinąd sympatyczny – zwierzak z wiersza Juliana Tuwima ma ze mną wiele wspólnego (i nie mam tu na myśli imienia). Wprawdzie nie jest aż tak źle, abym zapominał jak się nazywam, ale... W natłoku spraw codziennych bardzo często zapominam o tym, co ma nastąpić. Wydarzenia, które zdają się być na początku bardzo odległe, pojawiają się nagle tu i teraz. Pół biedy, jeśli jest to rzeczywiście *tu i teraz*, bo zwykle można jeszcze coś zrobić. A jeżeli jest to *tam i wczoraj*?

Poza tym, prócz wydarzeń, które po prostu pojawiają się na linii czasu (urodziny, imieniny, rocznice itp.), w naszym życiu mamy często do czynienia z tzw. terminami. Data i czas oznaczają wtedy miejsce, w którym kończy się okres na wykonanie jakiejś czynności (zadania) i musimy pochwalić się efektem końcowym. Za czasów szkoły podstawowej, a nawet średniej taką, nie rozłożoną wcześniej w czasie, pracę można było wykonać *na kolanie* (komu z nas nie zdażyło się odrabiać zadania domowego na przerwie, tuż przed lekcją?). W dorosłym życiu takie rzeczy się już nie udają, a w dodatku konsekwencje naszej beztroski są już nie tylko przykre, ale – nie rzadko – bardzo poważne.

Znając historię zapominalskiego słonia możemy przypuszczać, że nawet gdyby zaczął ćwiczyć na swojej trąbie pismo węzełkowe (co byłoby chyba wyjątkowym sposobem robienia notatek), to i tak nic by to nie dało. Na szczęście rzeczywistość wygląda zupełnie inaczej, mamy bowiem wiele, czasem zadziwiająco skutecznych, sposobów na poprawienie swojej pamięci. W tym artykule skupimy się jednak na tym, jak bezpiecznie uwolnić się od pamiętania konstruując (lub wykorzystując już istniejące) przypominacze. Innymi słowy, zajmiemy się sposobami na... zapominanie. Wcześniej musimy tylko odpowiedzieć sobie na jedno, ważne pytanie.

2.2. O czym możemy zapomnieć?

Do zapamiętania mamy wiele, są jednak takie sprawy, które nie muszą absorbować naszej uwagi do chwili, aż nie zbliży się ich termin. Spróbujmy zatem wyszczególnić takie kategorie zdarzeń i przyjrzyjmy się im bliżej.

Zdarzenia jednorazowe: wszystko , co wydarzy się tylko raz, albo ma miejsce w nieregularnych odstępach czasu (zdarza się raz kiedyś). Może być to coś, co nastąpi już niedługo (spotkanie z kolegą/koleżanką, wizyta u dentysty, wyjście do kina, mecz w telewizji), ale może to być również termin dość odległy – kilka miesięcy czy lat (np. ślub przyjaciela z Twoją obecną żoną, choć o tym dowiesz się raczej później, niż wcześniej).

Zdarzenia cykliczne: jeśli staramy się jakoś zapanować nad czasem (planujemy?) to takich zdarzeń jest chyba najwięcej. Niestety, ich główna cecha, czyli powtarzalność, wcale nie pomaga w pamiętaniu o nich. Ta kategoria jest zresztą bardzo obszerna i zawiera wiele „podkategorii". Znajdziemy tu zarówno sprawy dość ważne, a więc wszelkiego rodzaju rocznice (urodziny, imieniny, rocznice ślubu itp.), terminy płatności (składki ubezpieczeniowe, opłaty lokalowe, raty kredytów itp.), jak też sprawy całkiem banalne czy rytunowe (zakupy, sprzątanie).

Wśród tych wszystkich spraw są też takie, które mogą należeć do jednej, albo drugiej kategorii. Możemy przecież planować stały odstęp między kolejnymi wizytami u fryzjera czy w myjni samochodowej, ale wcale nie musimy tego robić. Ale nawet wtedy, gdy planowanie jakiejś czynności jest potrzebą chwili, w naszym kalendarzu wydarzeń pojawia się kolejny termin, o którym należy pamiętać.

A czy pamiętanie o czasie takich zdarzeń nie jest przypadkiem całkowicie zbędnym balastem? I czy nie jest tak, że jeśli termin jest odległy i/lub sprawa ma niski priorytet, nasza pamięć zwykle zawodzi?

Myślę, że teraz, kiedy stwierdziliśmy już w czym tkwi problem, pozostało tylko znaleźć skuteczne sposoby na uporanie się z nim **raz na zawsze**.

I tutaj właśnie zaczyna się nasza przygoda, ponieważ zawsze wtedy, gdy wiemy dokładnie czego chcemy, z pomocą przychodzi nam Linux.

3. Automaty małe i duże

Większość zadań, o których wcześniej wspomnieliśmy daleko wykracza poza zakres możliwości naszego systemu. Linux nie pójdzie przecież za nas do fryzjera, chociaż... może starszy o kolejne dziesięć lat będzie już w stanie sam nas ostrzyc.

Istotne dla nas jest to, że z automatyzacją zadań, niejednokrotnie bardzo złożonych, radzi sobie wręcz doskonale. W dodatku jest systemem bardzo przejrzystym, wydajnym i stabilnym. To wszystko sprawia, że świetnie nadaje się do tego, aby zrzucić na niego obowiązek pamiętania i przypominania nam o wszystkim, o czym (nie) musimy pamiętać.

Na początek zajmiemy się narzędziami, które każdy z nas znajdzie w swojej ulubionej dystrybucji. Potem zobaczymy, jak w zapominaniu mogą nam pomóc inne programy.

3.1. At

Program przeznaczony jest do uruchamiania pojedynczych zadań w późniejszym terminie. Ciekawe jest to, że sam termin może być określony na różne sposoby. Możemy podać dokładny dzień i godzinę, ale możemy też powiedzieć programowi, aby zrobił coś w samo południe, jutro czy np. za 3 godziny. `At` przyjmuje polecenia pisane bezpośrednio...

```
tskrynnyk@zgul:~$ at now+3hours
warning: commands will be executed using /bin/sh
at> echo -e "\a Idź do domu!" > /dev/tty4
at> <EOT>
job 46 at 2004-07-07 17:34
```

... oraz takie, które znajdują się w osobnym pliku.

```
tskrynnyk@zgul:~$ at -f przypomnij.sh teatime
warning: commands will be executed using /bin/sh
job 47 at 2004-07-07 16:00
```

Daje nam to możliwość wykorzystania go do przypominania o terminach, które określiliśmy wcześniej, jako jednorazowe. Wystarczy stworzyć skrypt, który wykonuje jakąś czynność i nakazać `at`, aby uruchomił go w ustalonym czasie.

Oczywiście program został stworzony do wykonywania polecenia o zadanym czasie i... tylko do tego, więc wykorzystywanie go w roli przypominacza wymaga kilku zabiegów. Przede wszystkim,

jak już wspomniałem, musimy mieć jakiś skrypt/program, w którym zawrzemy faktyczną reakcję systemu na zdarzenie. Poza tym, mimo wielu sposobów na określenie czasu, nie mamy możliwości wyznaczenia terminu, który jest pojęciem typowo kalendarzowym (*pierwszy piątek przyszłego miesiąca* jest dla `at` całkowitą abstrakcją). Jednak nawet w kontekście naszych rozważań te cechy wcale nie muszą być wadami. Właśnie dzięki swej prostocie `at` może być bardzo pomocny, jako element większego systemu. Możemy go przecież „podpiąć" do dowolnego programu, wywoływać z wnętrza skryptu, albo jeszcze inaczej łączyć z osobnymi narzędziami. Warto o tym pamiętać.

Program `at` jest standardowym składnikiem systemu. Jego autorem jest Thomas Koenig.

3.2. Cron

Ten program, podobnie jak `at`, jest narzędziem do wykonywania przyszłych zadań, więc do przypominania możemy wykorzystywać go na różne sposoby. Od tego pierwszego różni się jednak zasadniczo, `cron` interesuje się bowiem tylko wydarzeniami cyklicznymi. I trzeba przyznać, że radzi sobie z nimi doskonale. Mamy tu możliwość określenia okresów na wiele różnych sposobów. Możemy kazać programowi podejmować akcję zawsze o tej samej godzinie, w konkretny dzień tygodnia czy miesiąca. Dość proste jest też tworzenie bardziej złożonych cykli, np. *zawsze o godz. 7:30, w każdy wtorek i czwartek, ale tylko od września do czerwca.*
Wystarczy w tym celu otworzyć naszą tablicę.

```
tskrynnyk@zgul:~$ crontab -e
```

I dokonać odpowiedniego wpisu, w naszym przykładzie będzie to:

```
30  7  *  1-6,9-12  2,4     /home/tskrynnyk/przypomnij.sh
```

Nasz systemowy `cron` pozwala nam, jak widać, bardzo łatwo i przyjemnie uwolnić się od pamiętania o wydarzeniach periodycznych. Musimy jednak wiedzieć, że tak jak w przypadku `at`, do dyspozycji mamy narzędzie, które bardzo dokładnie wie *co* i *kiedy* należy wykonać, ale nie ma zielonego pojęcia o przypominaniu – znów potrzebujemy zewnętrznego programu. Poza tym nie rozumie czasu określonego, jako np. *każdy drugi poniedziałek miesiąca*, nie radzi sobie więc i z bardziej wyrafinowanymi warunkami, czyli np. *każdy ostatni dzień miesiąca, ale jeśli to akurat dzień wolny (weekend) to pierwszy dzień po nim*. Nic nie stoi jednak na przeszkodzie, aby `cron` uruchamiał codziennie skrypt, który będzie na początku sprawdzał czy to aby nie dzisiaj jest właśnie ten dzień, w którym należy coś zrobić. Przykładem niech będzie *ostatni dzień miesiąca* i to bez dodatkowych warunków, bo określenie takiego terminu też już przerasta `crona`.
Najprostszym sposobem stwierdzenia czy ostatni dzień miesiąca przypada dzisiaj jest porównanie „dzisiejszego" miesiąca z „jutrzejszym", czyli sprawdzenie czy dzisiaj i jutro jest taki sam miesiąc. Wykorzystamy do tego systemowy `date` i wbudowane w powłokę polecenie `test`. Początek takiego skryptu może wyglądać następująco:

```
DZIS=‘date %b‘
JUTRO=‘date -d ’1 day’ %b‘

[ $DZIS != $JUTRO ] || exit 0

# I dalej to, co należy zrobić,
# jeśli to ostatni dzień m-ca
...
```

Teraz wystarczy tylko nakazać `cronowi`, aby uruchamiał nasz skrypt w odpowiednim czasie. Celowo nie napisałem „codziennie", ponieważ przy drobnych zmianach taki skrypt może sprawdzać inne warunki. Szukanym dniem może być np. rzeczony *pierwszy poniedziałek miesiąca*, wtedy skrypt będzie uruchamiany tylko raz na tydzień.

Jak na zwykłe narzędzie systemowe, `cron` posiada ogromne możliwości. Sposobów na ich wykorzystanie może być bardzo wiele. Potrzebny jest „tylko" pomysł.

Program `cron` jest standardowym składnikiem systemu. Jego autorem jest Paul Vixie.

3.3. Anacron

Opisane dotychczas narzędzia dobrze spełniają swoją rolę w sytemie i pomogą zapominalskim pod jednym warunkiem – komputer musi być włączony w tym czasie, w którym zostały zaplanowane jakieś zadania, inaczej nie zostaną wykonane. W przypadku maszyn produkcyjnych (serwerów) problem taki nie istnieje, ale co mają zrobić użytkownicy komputerów domowych? Odpowiedzią jest program `anacron`. To on dba o to, aby wszystkie zaległe sprawy zostały załatwione zaraz po włączeniu komputera. Sprawdź czy masz go w systemie.

Pierwszym autorem programu `anacron` jest Christian Schwarz, ponownie przepisał go i zaimplementował Itai Tzur.

3.4. Cal/Ncal

Program `cal` (w wersji FreeBSD – `ncal`) nie jest oczywiście żadnym automatem, pojawił się jednak tutaj nie przypadkowo. Efektem jego działania jest tylko prosty kalendarz...

```
tskrynnyk@zgul:~$ ncal
 Lipiec 2004
po    5 12 19 26
wt    6 13 20 27
śr    7 14 21 28
cz 1  8 15 22 29
pi 2  9 16 23 30
so 3 10 17 24 31
ni 4 11 18 25
```

... ale już taka wizualizacja miesiąca może być (i dla mnie często jest) bardzo pomocna. Program potrafi m.in. pokazać nam konkretny miesiąc, rok i dodać do tego jeszcze numery tygodni. Ważne jest też to, że bardzo łatwo możemy z jego pomocą otrzymać widok, w którym np. będą zaznaczone konkretne daty. Bardzo przydatne narzędzie do prostych zastosowań.

Program `cal` jest standardowym składnikiem systemu. Jego autorem jest Wolfgang Helbig.

3.5. Calendar

Pierwszym programem, z którym zetkniemy się poszukując w naszym systemie przypominaczy z prawdziwego zdarzenia będzie właśnie `calendar`. Jest to wyjątkowo przyjemne narzędzie dla wszystkich tych, którzy chcą mieć dostęp do informacji o typowych rocznicach (urodziny, imieniny, wydarzenia historyczne etc.). Program jest dostarczany razem z przygotowanymi już kalendarzami. Są to pliki zawierające informacje o wydarzeniach z danej dziedziny (daty wraz z krótkim opisem). Znajdziemy tu wydarzenia ze świata muzyki, historii powszechnej, czy daty urodzin/śmierci znanych

ludzi[2]. Wystarczy krótka konfiguracja i wszystkie interesujące nas informacje otrzymamy na skrzynkę pocztową (jest to działanie domyślne) i po wydaniu polecenia `calendar`. Jeśli zdecydowaliśmy się przykładowo na informacje dotyczące komputerów, historii i muzyki[3] to 7 lipca wynik programu może być taki:

```
tskrynnyk@zgul:~$ calendar
 8 lip* Bell Telephone Co. formed (predecessor of AT&T), 1877
 8 lip* CDC incorporated, 1957
 7 lip* Urodził się Ringo Starr (Richard Starkey), Liverpool, Anglia, 1940
 7 lip* First radio broadcast of "Dragnet", 1949
 8 lip* First public reading of the Declaration of Independence, 1776
 8 lip* Liberty Bell cracks while being rung at funeral of John Marshall, 1835
```

Jak widać, `calendar` pokazując wydarzenia domyślnie „zagląda" też do dnia jutrzejszego. Konfiguracja programu i odpowiednie opcje przy wywołaniu pozwalają jednak wpłynąć na wiele rzeczy dotyczących jego zachowania.

Nie jesteśmy oczywiście ograniczeni tylko do gotowych zestawień. Możemy tworzyć własne i korzystać tylko z nich, albo łączyć oba źródła. Co więcej, program pozwala nam na uwzględnienie w naszym kalendarzu wydarzeń cyklicznych, a na dodatek można je określać na kilka sposobów. Może być to np. *każdy wtorek*, *każdy 1 lipca*, *15 każdego miesiąca*, *ostatnia sobota kwietnia*, *drugi piątek maja*... i nie jest to nawet specjanie trudne w zapisie.

```
Thursday    Każdy wtorek
June        Każdy 1 lipca
15 *        15 każdego miesiąca
04/SunLast  Ostatnia sobota kwietnia
May Fri+2   Drugi piątek maja
```

Program nie jest może szczytem marzeń wymagającego Trąbalskiego, ale swoją rolę spełnia bardzo dobrze. A jeśli przyjrzymy się mu bliżej i poznamy dodatkowe możliwości (m.in. przeglądanie wydarzeń w danym dniu/okresie), przyznać musimy, że `calendar` wart jest uwagi. Sam poznałem i polubiłem go już dawno i do dziś wykorzystuję z powodzeniem.

Program `calendar` jest standardowym składnikiem systemu.

3.6. Vim

`Vim`, czyli Vi IMproved jest – jak wiadomo – edytorem, jeśli jednak pracujemy z nim na codzień, wiemy doskonale o tym, że z pomocą odpowiednio przygotowanych skryptów możemy rozszerzać jego funkcjonalność. Takim dodatkiem może być właśnie kalendarz wywoływany z poziomu programu, w którym możemy nawet robić notatki przyporządkowane do danego dnia. Tę dodatkową funkcję programu zawdzięczamy Yasuhiro Matsumoto, bo to on wpadł na ten ciekawy pomysł i stworzył skrypt `calendar`. W działaniu jest to trochę połączenie systemowego `cal` i `calendar`. Mamy tu dostęp do widoku kalendarza na kolejne miesiące i w łatwy sposób możemy zapisywać zdarzenia na każdy dzień. Data, do której została przypisana jakaś notatka będzie potem specjalnie wyróżniona. Skrypt jest bardzo przyjemny w obsłudze, łatwo się instaluje, konfiguruje i lokalizuje. Właściwie to wystarczy zrobić odpowiednie wpisy w swoim `~/.vimrc`.

```
let g:calendar_mruler = 'Sty,Lut,Mar,Kwi,Maj,Cze,Lip,Sie,Wrz,Paź,Lis,Gru'
```

[2] Gotowe kalendarze są głównie w języku angielskim.

[3] Mój prywatny kalendarz muzyczny jest po polsku i dotyczy głównie jazzu, w oryginalnym króluje rock 'n' roll.

```
let g:calendar_wruler = 'Ni Po Wt Śr Cz Pt So'
let g:calendar_navi_label = ',Dzisiaj,'
let g:calendar_monday = 1
```

Osobiście nie wykorzystuję funkcji robienia notatek, ale ponieważ bardzo intensywnie używam `vima`, dzięki skryptowi `calendar` zawsze mogę rzucić okiem na kalendarz.

Opis dotyczy `vima` w wersji 6.0, jego głównym autorem jest Bram Moolenaar.
Program jest dostępny pod adresem:
http://www.vim.org
Autorem skryptu `calendar` jest Yasuhiro Matsumoto.
Skrypt dostępny jest pod adresem:
http://vim.sourceforge.net/scripts/script.php?script_id=52

3.7. Emacs

Kolejny znany edytor, `emacs`, pamiętnik i kalendarz ma już w sobie, wystarczy tylko odpowiednio go skonfigurować. Program potrafi m.in. pokazywać datę w różnych formatach, obliczyć godzinę wschódu i zachodu słońca oraz fazę księżyca. Pozwala też na prowadzenie pamiętnika (dziennika), a także daje się wykorzystywać jako typowy przypominacz. To wszystko robi wrażenie i działa znakomicie.

Wszelkie ustawienia dot. `emacsa` wpisujemy do swojego `~/.emacs`. Takie dodatkowe ustawienia pozwalają ustalić praktycznie wszystko, co dotyczy przypominacza, kalendarza, strefy czasowej, formatu daty itp. Nie będę opisywał wszystkiego po kolei, tylko pokażę kawałek mojego pliku.

```
(setq european-calendar-style t)
(setq display-time-24hr-format t)
(display-time)
(setq calendar-week-start-day 1)
(setq calendar-latitude 52.14)
(setq calendar-longitude 21.0)
(setq calendar-location-name "Warszawa")
(setq calendar-date-string '(year "-" month "-" day))
(setq calendar-day-name-array ["Niedziela" "Poniedziałek" "Wtorek" \
    "Środa" "Czwartek" "Piątek" "Sobota"])
(setq calendar-month-name-array ["Styczeń" "Luty" "Marzec" "Kwiecień" \
    "Maj" "Czerwiec" "Lipiec" "Sierpień" "Wrzesień" "Pzaździernik" \
    "Listopad" "Grudzień"])
```

Podobnie, jak w przypadku `vima`, dodatkowe możliwości edytora przydają się najbardziej tym, którzy z niego korzystają. `Emacs` jest jednak potężnym programem (właściwie to chyba już samodzielnym środowiskiem) i te dodatkowe funkcje są tu bardzo rozbudowane. Dość wspomnieć, że program potrafi nie tylko wyświetlać kalendarze i to z wyszczególnieniem świąt czy własnych wpisów, ale może nawet wyeksportować je do LaTeX-a (wychodzą z tego bardzo przyjemne wydruki).

Opis dotyczy `emacsa` w wersji 20.7.2, jego autorem jest Richard Stallman.
Program jest dostępny pod adresem:
http://www.gnu.org/software/emacs/

3.8. Remind

`Remind` jest programem, o którym można napisać osobny artykuł. W dołączonej dokumentacji możemy już na wstępie przeczytać, że `remind` to *sophisticated reminder service* i jest to prawda. To rzeczywiście *wyrafinowany* przypominacz posiadający tak wiele ciekawych (nie tylko z punktu widzenia Trąbalskiego) możliwości, że aż trudno się zdecydować, o których wspomnieć. I to może być chyba jedyną wadą tego programu. Możliwości ma tak wiele, że czasem trudno to wszystko ogarnąć (sam manual to mała książeczka).

Przede wszystkim prócz zwykłego przypominania o wydarzeniach i terminach określonych, jako daty i godziny, `remind` sam potrafi – w oparciu o podane informacje – wyliczyć którego dnia i o której godzinie będzie miało miejsce dane zdarzenie. Oczywiście nie mówię tu o prostych kalkulacjach typu *pierwszy poniedziałek miesiąca* czy *w każdy czwartek o godz. 16:30 w lipcu i sierpniu*. Ten program potrafi zdecydowanie więcej. Oprócz tego, że – tak jak `emacs` – umie policzyć wschody i zachody słońca czy fazy księżyca, potrafi też sam określić na jaki dzień przypada np. nasza wypłata uwzględniając przy tym (przykry) fakt, że dostaniemy ją później, jeśli *pierwszego* to akurat sobota, albo niedziela. Potrafi też, dzieki wbudowanemu językowi, wykonać inne wymyślne kalkulacje oraz dynamicznie konstruować treść przypomnień i to w zależności od tego, w jakiej formie prezentowana będzie informacja o terminach. A do tego jest wielojęzyczny, może wyświetlać kalendarze na wiele różnych sposobów i jeszcze eksportować je do PostScriptu. Warto też wspomnieć o tym, że `remind` pracując w trybie daemona może nie tylko przypominać, potrafi również uruchamiać inne programy. Dzięki temu, zamiast przypominać o tym, że mamy coś do zrobienia, o wyznaczonej porze sam się tym zajmie.

Konfiguracja `reminda` polega na umieszczeniu w pliku `~/.reminders` odpowiednich instrukcji, które będą dotyczyć zarówno ustawień programu, jak i terminów przypomnień. Możemy zatem umieścić tu np. informację o naszej strefie czasowej (niezbędne do prawidłowych wyliczeń m.in. wschodów/zachodów słońca).

```
SET $CalcUTC 0
SET $MinsFromUTC -300
```

Ustawić wygląd przypomnień pojawiających się na konsoli (format, kolory). Zdecydować o wyglądzie kalendarzy w formacie PostScript. Wpisać własne funkcje... ale zacznijmy może od rzeczy najprostszych, czyli terminów.

Format zapisu wydarzeń jest dość złożony, ale prosty wpis dotyczący np. jakiegoś święta może wyglądać po prostu tak:

```
8 Marzec Dzień Kobiet
```

Kiedy poznamy bliżej program, taki wpis będzie jednak wyglądał trochę inaczej, np.:

```
REM 8 Mar 7 MSG %''Dzień Kobiet%'' %b.%
```

Teraz będziemy m.in. powiadamiani już na tydzień przed terminem, a komunikat będzie informował dodatkowo o tym, ile dni pozostało do tego święta.
Bardziej złożone terminy to chociażby wspomniana już wypłata, a więc *pierwszego, albo zaraz po pierwszym*.

```
REM 1 3 AFTER OMIT Sobota Niedziela MSG %''Wypłata%'' %b.%
```

Dla przesądnych mam jeszcze inny przykład – ostrzeżenie o trzynastym w piątek. Zdarza się to niezbyt często, ale dzięki `remindowi` możemy wiedzieć o tym już np. na miesiąc przed feralnym dniem.

```
REM 13 30 SATISFY [wkdaynum(trigdate()) == 5] MSG %''Black Cat!%'' %b.%
```

Skoro już jesteśmy przy poważniejszych kalkulacjach zademonstruję jeszcze sposób na umieszczanie wyników takich wyliczeń bezpośrednio w komunikacie.

Chcemy przykładowo, aby w przypomnieniach o urodzinach naszych przyjaciół czy członków rodziny było widać, która to właściwie rocznica. W tym celu piszemy małą funkcję, która pobiera rok urodzenia i zwraca liczbę przeżytych lat.

```
FSET ilemalat(x) year(trigdate())-x
```

Umieszczamy ją gdzieś na początku naszego `~/.reminders`, a potem wykorzystujemy do sformatowania komunikatu przypomnienia.

```
REM 25 Sie +7 MSG %''[ilemalat(1991)]. urodziny Linuksa%'' %b.%
```

To wszystko. Na tydzień przed terminem zaczniemy być informowani o nadchodzącej rocznicy. Jeśli chcielibyśmy już teraz zobaczyć efekt działania takiego zapisu wystarczy, że uruchomimy program z odpowiednią opcją.

```
tskrynnyk@zgul:~$ rem -t
Przypomnienia na dziś...
 ...
13. urodziny Linuksa za 49 dni.
 ...
```

Takich opcji wywołania mamy zresztą sporo...

```
tskrynnyk@zgul:~$ remind
REMIND 03.00.22 (Polish version) Copyright 1992-1998 David F. Skoll
Copyright 1999-2000 Roaring Penguin Software Inc.

Sposób użycia: remind [opcje] plik [data] [czas] [*powtórzenie]
Opcje:
 -n     Wypisz następne przypomnienia w prostym formacie
 -r     Zablokuj dyrektywy RUN
 -c[n]  Wypisz kalendarz na n (domyślnie 1) miesięcy
 -c+[n] Wypisz kalendarz na n (domyślnie 1) tygodni
 -w[n[,p[,s]]]  Ustaw szerokość, wypełnienie i odstępy w kalendarzu
 -s[+][n] Wypisz uproszczony kalendarz na n (1) miesięcy (tygodni)
 -p[n]  To samo co -s, ale kompatybilne z rem2ps
 -v     Obszerniejsze komentarze
 -o     Ignoruj instrukcje ONCE
 -t     Odpal wszystkie przyszłe przypomnienia niezależnie od delty
 -h     Praca bezszmerowa
 -a     Nie odpalaj przyponień czasowych - kolejkuj je
 -q     Nie kolejkuj przyponień czasowych
 -f     Nie przechodź do pracy w tle
 -z[n]  Pracuj jako demon, budząc się co n (5) minut
 -d...  Odpluskwianie: e=echo x=expr-eval t=trig v=dumpvars l=showline
 -e     Komunikaty o błędach skieruj na stdout
 -b[n]  Format czasu: 0=am/pm, 1=24godz., 2=żaden
 -x[n]  Limit powtórzeń klauzuli SATISFY (domyślnie=150)
```

```
-kcmd  Wywołaj 'cmd' dla przypomnień typu MSG
-g[ddd] Sortuj przypomnienia według daty, czasu i priorytetu
-ivar=val Zainicjuj zmienną var wartością val i zachowaj ja
-m     Rozpocznij kalendarz od poniedziałku zamiast od niedzieli
```

Nie wspomniałem jeszcze o wielu innych rzeczach (m.in. o organizacji czasu, o pracy w trybie daemona), ale wydaje mi się, że już teraz widać wyraźnie, jak ogromne możliwości ma `remind`. Polecam gorąco, ten przypominacz jest naprawdę *sophisticated*.

Opis dotyczy programu `remind` w wersji 03.00.22, jego autorem jest David F. Skoll.
Program jest dostępny pod adresem:
http://www.roaringpenguin.com/penguin/open_source_remind.php

4. Zrób sobie własny automat

To, jak wykorzystamy opisane tutaj programy zależy tylko od indywidualnych potrzeb i inwencji. Mogą to być proste rozwiązania, np. własny skrypt „wrzucony" do `crona`, ale też bardziej zaawansowane systemy wykorzystujące wiele narzędzi systemowych i programów zewnętrznych. Każde z nich, jeśli dobrze przemyślane, będzie pomocne i będzie nam (a może również innym) służyć przez wiele lat. Zacznijcie już teraz tworzyć coś praktycznego, ale wcześniej zobaczcie jeszcze kilka prostych przykładów.

4.1. Beep co godzinę

Informacja o upływającym czasie jest bardzo ważna, szczególnie wtedy, gdy czas ten spędzamy siedząc przed komputerem. Nie mówię oczywiście o zegarku umieszczonym gdzieś na pulpicie, ani o konsolowym `vcs`, ponieważ bez dodatkowego bodźca mogę przez wiele godzin nawet na nie nie spojrzeć. Takim bodźcem może być jakiś sygnał dźwiękowy pojawiający się, powiedzmy, o każdej pełnej godzinie. Miałem już do czynienia z różnymi programami, które taką informację przekazywały, są wśród nich nawet takie, które po prostu mówią, która jest godzina. Dla mnie, z wielu powodów, najbardziej odpowiedni jest zwykły *beep* z głośniczka.
Jeśli wiemy już, co chcemy osiągnąć, realizacja jest banalnie prosta.

Do generowania sygnałów przez sprzętowy głośnik wykorzystuję napisany przez Johnathana Nightingale'a program `beep`. Stosuję go też do innych celów, stąd mój wybór. `Beep` potrafi sporo, ale tutaj wystarczą mi trzy bardzo krótkie sygnały o wysokiej częstotliwości:

```
beep -f 1760 -D 30 -l 50 -r 3
```

Teraz wystarczy wpisać do swojej tablicy `crona` zadanie wykonania powyższego polecenia zawsze o każdej pełnej godzinie:

```
0  * * * *  test -x /usr/bin/beep && beep -f 1760 -D 30 -l 50 -r 3 >/dev/null
```

I nasz zegarek jest gotowy.
Ja mam u siebie jeszcze...

```
30 * * * *  test -x /usr/bin/beep && beep -f 440 -l 50 >/dev/null
```

... ale taki rygor polecam tylko tym, którzy naprawdę tego potrzebują.

Ten przykład pokazuje, że nawet to, co pełni ważną rolę wcale nie musi być skomplikowane. Wręcz odwrotnie – może być czytelne, rozszerzalne i uniwersalne tak bardzo, jak tylko się da.

Autorem programu beep jest Johnathan Nightingale.
Program jest dostępny pod adresem:
http://johnath.com/beep/

4.2. Kartka z kalendarza

Może się zdarzyć, że mając konto na serwerze, gdzie znajduje się calendar, możemy z niego korzystać tylko przez bezpośrednie wywołanie w powłoce. Mimo tego, że w swoim katalogu domowym mamy katalog ~/.calendar, a w nim nawet plik z prywatnymi wydarzeniami, calendar nie wysyła do nas żadnych listów. Przyczyną jest pewnie to, że administrator nie zdecydował się na codzienne uruchamianie programu w taki sposób, aby rozsyłał pocztę do wszystkich użytkowników, którzy tego chcą. W takiej sytuacji musimy radzić sobie sami. Wykorzystamy do tego mail, calendar oraz cron. Załóżmy, że zdecydowaliśmy się na otrzymywanie informacji tylko z dnia dzisiejszego (ja właśnie tak wolę), a więc bez domyślnego *look ahead*. Program powinien być zatem uruchomiony z odpowiednią opcją:

```
calendar -l0
```

Teraz trzeba to, co zwróci tak wywołany calendar przesłać, jako treść e-maila na naszą skrzynkę, załatwi to zwykły potok:

```
calendar -l0 | mail -s 'Zdarzyło się...' tskrynnyk@zgul.ime.uz.zgora.pl
```

Pozostało już tylko wpisać tę komendę do naszego crontaba:

```
0 6 * * *  test -x /usr/bin/calendar && calendar -l0 | \
   mail -s 'Dzisiaj wydarzyło się...' tskrynnyk@zgul.ime.uz.zgora.pl >/dev/null
```

i... o całej sprawie jak najszybciej zapomnieć.

4.3. Terminarz inaczej

Bardzo dobrym przykładem własnego rozwiązania jest też terminarz, którego autorem jest Cezary M. Kruk. Jest to skrypt pokazujący kalendarz i terminarz korzystający z plików tekstowych, w których – podobnie, jak w przypadku calendar – znajdują się wpisy dot. wydarzeń. Skrypt jest o tyle ciekawy, że mimo tego, iż jest dość rozbudowany, wykorzystuje tylko to, co każdy z nas ma w systemie, a więc znany nam już cal oraz date, cut etc. Myślę, że to świetna demontracja tego, jak elastyczny i uniwersalny jest nasz system.
A skoro już mowa o uniwersalności, może dodam tutaj coś od siebie.

Już przy opisie programu cal wspominałem o tym, że występuje on w dwóch wersjach. Różnią się one tym, że jedna wyświetla tygodnie w pionie, a druga w poziomie. W dodatku opcja -m, która mówi cal, że tydzień ma się rozpoczynać od poniedziałku, nie występuje w ncal, więc wywołanie go z tym przełącznikiem powoduje tylko błąd. Zauważyliście zapewne, że w moim systemie znajduje się ncal.
Aby uczynić terminarz bardziej przenośnym, wystarczy na początku skryptu zadeklarować zmienną, w której każdy będzie mógł wpisać odwołanie do programu, który ma w swoim systemie. W moim przypadku będzie to zatem:

```
my_cal=/usr/bin/ncal
```

Potem wszystkie sztywne wywołania /usr/bin/cal -m, należy zamienić na $my_cal i po kłopocie.

Nie będę tu omawiał całego programu, ponieważ jest to prosty w analizie skrypt powłoki. Rozprowadzany jest zresztą razem z gotowym plikami-bazami imieninowymi i świątecznymi, a nawet własną stroną man. Pokaże tylko – dla zachęty – jak się prezentuje.
Wywołany bez opcji...

```
tskrynnyk@zgul:~$ terminarz-pl
Lipiec 2004
po     5 12 19 26
wt     6 13 20 27
śr    <7>14 21 28
cz  1  8 15 22 29
pi  2  9 16 23 30
so  3 10 17 24 31
ni  4 11 18 25

śro lip 7 2004
lip  7 Estery
lip  7 Ewalda
```

... oraz w celu odnalezienia konkretnego wpisu...

```
tskrynnyk@zgul:~$ terminarz-pl Zorn
2002 lis 16 John Zorn w Katowicach
```

Zanim sam zacząłem pisać tego typu skrypty, do zapominania używałem z powodzeniem właśnie `terminarza`. Szczerze polecam.

Autorem programu `terminarz` jest Cezary M. Kruk.
Program jest dostępny pod adresem:
http://c.kruk.webpark.pl#terminarz

5. Na zakończenie

Tak typowe w Linuksie łączenie małych narzędzi w celu zrobienia czegoś większego sprawdza się w wielu sytuacjach. Każdy, kto tego spróbował wie, że takie podejście do problemu jest skuteczne, ekonomiczne, a przy tym bardzo naturalne. W naszym systemie większość codziennych zadań realizowanych jest właśnie w taki sposób. Dlaczego mielibyśmy rezygnować z tego, aby komputer robił jeszcze inne pożyteczne rzeczy – specjalnie dla nas? Tym bardziej, że nawet jeśli już istnieją jakieś rozwiązania to przecież zawsze można (a czasem nawet trzeba) zrobić po swojemu, aby było dokładnie tak, jak chcemy. Takie małe, osobiste *scyzoryki* zwykle nie wymagają od nas zaawansowanej wiedzy i żadnych specjalnych zabiegów. Często wystarczy sprawne posługiwanie się ogólnie dostępnymi narzędziami (`sed`, `cut`, `sort`, `grep`, `date` itp.), aby uzyskac efekt, który będzie nas w pełni satysfakcjonował.

Mam nadzieję, że przynajmniej część z Was przekonałem do tego, że z pomocą Linuksa o wielu sprawach możemy zapomnieć **raz na zawsze**.

Skorowidz