Temat zajęć	Prawa plików. Narzędzia systemowe c.d.
Zakres materiału	Polecenia: `chgrp, chmod, chown, cmp, date, df, diff, du, file, ln, sleep, sync`

Prawa dostępu

W systemach klasy UNIX dostęp do plików i katalogów (oraz innych obiektów, które obsługiwane są tak, jakby były plikami) kontrolowany jest przez tzw. *prawa dostępu*, które regulują zasady na jakich użytkownicy mogą korzystać z tych zasobów. Wyróżnia się trzy podstawowe rodzaje dostępów:

- odczyt – oznaczane jako **r** (ang. *read*)
- zapis – oznaczane jako **w** (ang. *write*)
- wykonanie – oznaczane jako **x** (ang. *execute*)

Każdy z powyższych typów dostępów jest dopuszczany bądź wykluczany niezależnie dla:

- użytkownika (ang. *user*), który jest właścicielem pliku lub katalogu (domyślnie właścicielem jest użytkownik, który utworzył dany plik lub katalog);
- użytkowników, którzy należą do tej samej grupy (ang. *group*), do której należy plik lub katalog (jest to tzw. *grupa właścicielska*)
- dla pozostałych (ang. *other*) użytkowników.

Interpretacja praw dostępu w przypadku plików jest następująca:

- **`r`** – umożliwia odczytanie zawartości pliku
- **`w`** – umożliwia zapis do pliku oraz jego usunięcie
- **`x`** – umożliwia uruchomienie pliku, gdy zawiera on kod wykonywalny, binarny albo tekst skryptu

Znaczenie praw zmienia się radykalnie, kiedy odnoszą się do katalogu:

- **`r`** – umożliwia odczytanie metadanych plików znajdujących się w katalogu (co pozwala np. skutecznie wykonać polecenie **`ls`** w tym katalogu)
- **`x`** – umożliwia uczynienie katalogu katalogiem bieżącym, czyli wykonanie takiego polecenia **`cd`**, w wyniku którego znajdziemy się w tym katalogu
- **`w`** – umożliwia tworzenie, usuwanie i przemianowywanie plików znajdujących się w katalogu, ale tylko wtedy, gdy jednocześnie posiada się uprawnienie **`x`**; bez uprawnienia **`x`** uprawnienie **`w`** jest pozbawione znaczenia.

Podstawowym sposobem zasięgania informacji o prawach jest użycie polecenia `ls` z przełącznikiem `-l` – oto przykład wyjścia wyprowadzanego przez to polecenie:

```
# ls -l
   drwx------ 15 adam students 4096 lip  6 13:27 ./
   drwxr-xr-x 54 adam students 4096 lip  6 11:20 ../
   -rwxr--r-x  2 adam students 4096 cze 23 13:32 abc.txt
```

Informacja o prawach wyświetlana jest według następującego schematu:

user			***group***			***others***		
`r`	**`w`**	**`x`**	**`r`**	**`w`**	**`x`**	**`r`**	**`w`**	**`x`**

Wystąpienie jednego z powyższych znaków na wskazanej pozycji oznacza istnienie danego prawa, znak ”–“ oznacza jego brak.

Zatem dla pliku *abc.txt* określone są następujące prawa:

- dla właściciela dostępne są wszystkie dostępy
- dla członków grupy *students* dostępny jest tylko odczyt
- dla pozostałych użytkowników odczyt i wykonanie.

Prawami dostępu można także operować stosując notację numeryczną, w której każde prawo jest traktowane jest jako **bit** (istnienie prawa jest *jedynką*, brak prawa *zerem*), a uzyskaną w ten sposób wartość binarną prezentuje się jako liczbę **ósemkową**. Skoro więc komplet praw opisywany jest dziewięcioma bitami, równoważna postać ósemkowa składać się będzie z trzech cyfr. Każda z cyfr ósemkowych uwidacznia prawa jednego rodzaju użytkowników (u, g, o) i tworzona jest jako suma następujących wartości:

- 4 – dostęp **r**
- 2 – dostęp **w**
- 1 – dostęp **x**

Wynika stąd, że prawa pliku *abc.txt* zapisuje się w tej notacji jako `745`.

Należy pamiętać, że wpływ na to, jak i przez kogo może być używany dany plik, ma również to, **do kogo plik należy** i do jakiej grupy właścicielskiej jest przypisany. System Linux udostępnia zestaw poleceń umożliwiający efektywne manipulowanie każdą z tych danych. Są to:

- `chmod` (ang. *change mode*) do zmiany praw dostępu do pliku
- `chown` (ang. *change owner*) do zmiany właściciela pliku (chociaż należy zaznaczyć, że polecenie to pozwala również zmieniać grupę pliku)
- `chgrp` (ang. *change group*) do zmiany grupy pliku

Polecenie *chmod*

`chmod` *`[przełączniki...] uprawnienia nazwa_pliku...`*

Zmienia prawa dostępu w sposób wskazany pierwszym argumentem wywołania na rzecz plików lub katalogów wskazanych drugim argumentem wywołania. Uwaga: z oczywistych powodów zmiana praw pliku nie zawsze jest możliwa – np. nie można w ten sposób przydzielić sobie uprawnień do cudzego pliku.

Specyfikacja uprawnień, jakie plik ma nabyć po wykonaniu polecenia `chmod`, możliwa jest na kilka sposobów, używanych zależnie od okoliczności. Pierwszy z nich posługuje się czytelnym (kwestia gustu, ale na pewno czytelniejszym od zapisu ósemkowego) formalizmem pozwalającym określić nadawane bądź odbierane prawa, a który można zapisać w skrócie jako

kto-jak-co

Kto wskazuje jakiej kategorii użytkowników dotyczy operacja:
u – właściciel
g – grupa
o – pozostali
a – wszyscy
(litery te mogą być łączone w dowolne zestawy, choć nie wszystkie zestawy mają sens praktyczny; możliwe jest również użycie pustego zestawu, w takim przypadku uznaje się, że jest on tożsamy z **a**)

Jak opisuje operację, która ma zostać wykonana:

+ – dodanie uprawnień do już istniejących
- – odebranie uprawnień z już istniejących
= – przypisanie uprawnień bez względu na to, jakie istniały wcześniej

Co specyfikuje uprawnienia, które są brane pod uwagę w czasie wykonywania operacji i używa się tutaj niepustego złożenia znaków **r**, **w** i **x**. Zestaw *kto-jak-co* może wystąpić więcej niż raz, ale wtedy kolejne zestawy muszą być rozdzielone przecinkami.

Oto przykładowe zlecenia z wykorzystaniem powyższej składni polecenia `chmod`:

`chmod u+w plik.txt`
dodaje prawo zapisu dla właściciela do pliku *plik.txt*

`chmod go-x skrypt`
usuwa prawo wykonywania pliku *skrypt* przez użytkowników z grupy właścicielskiej i innych

`chmod a=r fb`
ustawia prawa dostępu do pliku fb na „*tylko do odczytu*” dla wszystkich użytkowników

`chmod u+x,g-x wirus1 wirus2`
ustawia prawa wykonywania plików *wirus1* i *wirus2* przez właściciela i odbiera to prawo grupie

Dygresja: zwróć uwagę, że prawo wykonania jest tak naprawdę czymś więcej, niż tylko zezwoleniem bądź jego brakiem. Można powiedzieć, że nadania tego prawa to *de facto* czyni plik wykonywalnym, jako że w odróżnieniu od systemów klasy Windows® informacja, czy o tym, czy plik jest wykonywalny nie pochodzi z jego nazwy. Oznacza to, że każdy plik (nawet bitmapa) może być z punktu widzenia systemu rozpoznawany jako wykonywalny. Sens takiego działania jest raczej dyskusyjny, ale jest to możliwe.
Polecenie `chmod` umożliwia także określanie praw dostępu w postaci ósemkowej, np.:

`chmod 777 file*`
ustawia wszystkie prawa (*róbta co chceta*) wszystkim użytkownikom do plików, których nazwy zaczynają się od liter *file*

`chmod 742 you`
ustawia prawa odczytu, zapisu i wykonywania właścicielowi, prawo odczytu użytkownikom z tej samej grupy oraz prawo zapisu innym użytkownikom do pliku *you*

Polecenie *chown*

`chown` *`[przełączniki...] właściciel[:grupa] plik...`*

Zmiana właściciela lub grupy pliku. Uwaga: nie każda zmiana właściciela jest dopuszczalna. Z całą pewnością nie możesz w ten sposób przywłaszczyć sobie pliku należącego do użytkownika *root*.

Polecenie *chgrp*

`chgrp` *`[przełączniki...] grupa plik...`*

Zmienia grupę, do której należy wskazany plik lub katalog. Obowiązują te same zastrzeżenia, jak przy poleceniu `chown`.

Użytkownicy i grupy – reprezentacja w systemie

W systemach klasy UNIX informacja o użytkownikach jest standardowo przechowywana w plikach tekstowych o znormalizowanej strukturze, nazwie i lokalizacji. Jeżeli konkretna instalacja tego wymaga, możliwe jest również stosowanie zamiast tego innych rozwiązań, np. w postaci systemów katalogowych lub baz danych, jednak nie są to rozwiązania typowe.

Pliki te to:

- `/etc/passwd`
- `/etc/shadow`
- `/etc/groups`

Pliki te mają strukturę wierszową, a każdy z wierszy jest dzielony na pola przy pomocy separatora, którym jest znak `:` (dwukropek).

`/etc/passwd`

Plik ten przeznaczony jest przede wszystkim do przechowywania informacji o:

- nazwie użytkownika
- identyfikatorze użytkownika (UID)
- położeniu katalogu domowego użytkownika
- informacji osobistych (nr telefonu, nr pokoju, etc)
- nazwy programu powłoki, który ma zostać uruchomiony po zalogowaniu użytkownika

Historycznie plik ten używany był również do przechowywania informacji o hasłach, co jest o tyle ciekawe, że domyślne prawa tego pliku to `644` (właścicielem jest użytkownik *root*). Oznacza to, że zawarte w pliku informacje są dostępne dla wszystkich (sic!). Oczywiście, w pliku tym przechowywano nie same hasła, a tzw. *hasze* haseł (aby uczynić historię krótszą, powiedzmy tylko, że są to wyniki przekształcenia haseł przy pomocy pewnej nieróżnowartościowej i nieodwracalnej funkcji; dodajmy, że funkcja ta jest znana publicznie). W momencie logowania system znajdował hasz z wprowadzonego hasła i porównywał go z haszem pamiętanym w `/etc/passwd`. Niestety, bardzo szybko okazało się, że wiara w to, że odtworzenie oryginalnego hasła z hasza nie jest możliwe, jest co najmniej naiwna. Praktyka pokazała, że oryginalne hasło wcale nie jest potrzebne – wystarczy odnalezienie tzw. *kolizji* tzn. ciągu znaków, który daje w wyniku taki sam hasz, a to otwiera na oścież podwoje systemu.

Narzucający się niemal natychmiast pomysł, że wystarczy zmienić prawa pliku `/etc/passwd` np. na `600` jest spóźniony – zbyt wiele narzędzi i usług zakłada, że plik ten jest publicznie odczytywalny. W tej sytuacji zdecydowano się na inne rozwiązanie: w pole przeznaczone na hasz hasła wpisuje się znak 'x' (który nie ma prawa wystąpić w pojedynkę w prawdziwym haszu hasła), a same hasze przechowuje się w pliku

`/etc/shadow`

który dla odmiany ma prawa `640` i którego nie da się już odczytać bez posiadania odpowiednio wysokich uprawnień.

`/etc/group`

Plik ten przeznaczony jest do przechowywania informacji o grupach użytkowników znanych w systemie. Dla każdej grupy pamięta się:

- jej nazwę

- jej identyfikator (GID)
- listę nazw użytkowników będących członkami grupy

Standardowe prawa tego pliku to `644`.

Dowiązania

W systemach klasy UNIX informacje o plikach na dysku przechowywane są w strukturach, które nazywa się *i-węzłami* (ang. *i-node*). Każdy taki i-węzeł przechowuje m. in. następujące informacje:

- prawa dostępu
- czas ostatniej zmiany zawartości lub metadanych pliku (ang. *ctime – last change time)* – sytuacja taka może zajść np. na skutek zmiany nazwy albo praw pliku, ale także po zapisie do pliku
- czas ostatniej modyfikacji zawartości pliku (ang. *mtime – last modification time*) – np. na skutek dopisania nowych danych na koniec pliku
- czas ostatniego dostępu do pliku (ang. *atime – last access time*) – dostępem takim jest najczęściej odczyt z pliku
- rozmiar pliku
- licznik dowiązań
- inne atrybuty

Zauważ, ze pomiędzy wymienionymi powyżej trzema czasami zachodzi następująca zależność:

- *atime* może się zmienić jako jedyny spośród tej trójki (np. na skutek odczytu z pliku)
- zmiana *ctime* automatycznie pociąga za sobą zmianę *atime* (np. po zmianie nazwy pliku)
- zmiana *mtime* wywołuje jednoczesną zmianę *atime* i *mtime* (np. po zapisie do pliku)

Warto dodać, że każdy z powyższych czasów reprezentowany jest w systemie plików jako dana całkowita bez znaku obrazująca liczbę sekund od północy czasu uniwersalnego 1 stycznia 1970 roku, co umownie uznaje się za początek tzw. epoki Uniksa (ang. *Unix Epoch*)

Licznik dowiązań określa **ile razy** dany plik jest dostępny w systemie plików (być może w różnych katalogach i pod różnymi nazwami). Licznik ten umożliwia realizację dowiązań do plików, które to dowiązania są dodatkowymi nazwami dla pliku, umożliwiającymi dostęp do oryginału (np. z poziomu różnych katalogów).

Istnieją dwa rodzaje dowiązań:

- dowiązania twarde (ang. *hard links*) realizowane fizycznie na poziomie organizacji i-węzłów wewnątrz systemu plików w sposób niewidoczny dla użytkownika
- dowiązania miękkie lub symboliczne (ang. *soft* lub *symbolic links*) realizowane poprzez pliki specjalnego przeznaczenia, reprezentujące dowiązanie w sposób logiczny

Dowiązania symboliczne mogą także dotyczyć katalogów oraz plików w innych systemach plików – informacja o nich dostępna jest dzięki omówionemu już poleceniu `ls -l`. Wszystkie dowiązania można przetwarzać dokładnie tak samo jak pliki zwykłe, w szczególności mogą także być usunięte poleceniem `rm`.

Tworzenie dowiązań jest możliwe dzięki poleceniu `ln` (ang. *link*), którego najczęściej stosowana postać prezentuje się jak poniżej:

```
ln [przełącznik...] plik_faktyczny nazwa_dowiązania
```

Argument *`plik_faktyczny`* musi wskazywać na istniejący plik (lub katalog w przypadku dowiązań symbolicznych), do którego tworzone jest dowiązanie, a drugim argumentem jest nazwa pod którą zostanie utworzone nowe dowiązanie. Utworzenie dowiązania symbolicznego wymaga zastosowania przełącznika `-s`. Przykładowe wywołania zlecenia utworzenia dowiązań:

`ln ./abc/plik.txt plik1.txt`
tworzy dowiązanie (twarde) do pliku *plik.txt* w katalogu *./abc* pod nazwą *plik1.txt* w katalogu bieżącym

`ln -s abc/plik.txt ~/plik1.txt`
tworzy dowiązanie symboliczne do pliku *plik.txt* w katalogu *abc* pod nazwą *plik1.txt* w katalogu domowym użytkownika.

Inne, rzadziej stosowanej, formy polecenia `ln` to:

`ln` *`[przełącznik...] plik_faktyczny`*
Tworzy dowiązanie w katalogu bieżącym o nazwie takiej, jaką nosi *plik_faktyczny*.

`ln` *`[przełącznik...] plik_faktyczny... katalog`*
Dla każdego z *plików_faktycznych* tworzy dowiązanie o takiej samej nazwie jak plik, ale we wskazanym *katalogu*.

`ln` *`[przełącznik...] -t katalog plik_faktyczny...`*
Działanie jak w poprzednim przypadku.

Dygresja: Ciekawą cechą dowiązań jest to, że jeśli pewien plik używany jest poprzez któreś swoich dowiązań, to traktowany jest tak, jakby nosił nazwę dowiązania, a nie nazwę pliku faktycznego. W szczególności, w programach napisanych w języku „C” parametr `argv[0]` będzie zawierał nazwę, której użyto, aby uruchomić dany program, a nie nazwę pliku, w którym znajduje się kod wykonywalny. Oznacza to, że ten sam kod może wypełniać różne funkcje w zależności od nazwy dowiązania, jakie spowodowało jego uruchomienie. Ta cecha dowiązań wykorzystywana jest np. przez program o charakterystycznej nazwie *busybox* (dołączany standardowo do praktycznie wszystkich dystrybucji Linuksa), który potrafi zachowywać się jak polecenia takie jak *cp*, *mv*, *rm* itp. o ile tylko zostanie uruchomiony taką właśnie nazwą. Rozwiązanie takie pozwala oszczędzić miejsce na nośniku w przypadku stosowania go w rozwiązaniach typu *live* dystrybuowanych na mediach o ograniczonej pojemności (np. na dyskietkach – tylko kto jeszcze używa dyskietek?) albo w systemach wbudowanych.

Polecenie date

Polecenie `date` pozwala każdemu użytkownikowi odczytać bieżący czas systemowy, a użytkownikowi o odpowiednio wysokich uprawnieniach (*root*) pozwala również tenże czas ustawić (czy potrafisz wyjaśnić dlaczego możliwość wykonania tej czynności przez każdego użytkownika byłaby z gruntu nierozsądna?). Z tego tez powodu omówimy tu wyłącznie te formy polecenia `date`, która może być użyta przez wszystkich:

`date` *`[przełączniki...] [+FORMAT]`*

- użyte bez argumentów wyświetli bieżący czas systemowy w formacie zgodnym z ustawioną „narodowością” systemu i z użyciem nazw miesięcy oraz dni tygodnia w stosownym języku naturalnym
- przełącznik `-R` spowoduje wyświetlenie czasu w formacie RFC 2822 (używanym np. w poczcie elektronicznej)
- przełącznik `-u` spowoduje wyświetlenie czasu przekształconego do czasu uniwersalnego (UTC)
- przełącznik `-r` z parametrem będącym nazwą pliku podaje czas ostatniej modyfikacji tego pliku

- parametr `FORMAT`, jeśli zostanie użyty, jest ciągiem znaków formatujących, podobnych w swojej postaci do tego, do czego przyzwyczaiła cię (mam nadzieję) funkcja `printf` z języka „C”; poniżej prezentujemy zaczerpnięty z manuala zestaw najważniejszych specyfikatorów używanych przez polecenie `date`:
  - `%a`
    lokalny skrót nazwy dnia tygodnia (np. `nie`)
  - `%A`
    lokalna pełna nazwa dnia tygodnia (np. `niedziela`)
  - `%b`
    lokalny skrót nazwy miesiąca (np. `sty`)
  - `%B`
    lokalna pełna nazwa miesiąca (np. `styczeń`)
  - `%c`
    lokalna data i czas (np. `czw mar 3 23:05:25 2005`)
  - `%C`
    stulecie, jak `%Y`, tylko z obciętymi ostatnimi dwiema cyframi
  - `%d`
    dzień miesiąca (np. 01)
  - `%D`
    data, to samo co `%m/%d/%y`
  - `%e`
    dzień miesiąca uzupełniony spacjami, jak `%_d`
  - %F
    pełna data; to samo co `%Y-%m-%d`
  - `%g`
    ostatnie dwie cyfry roku numeru tygodnia ISO (patrz %G)
  - `%G`
    rok numeru tygodnia ISO (patrz `%V`)
  - `%H`
    godzina (00..23)
  - `%I`
    godzina (01..12)
  - `%j`
    dzień roku (001..366)
  - `%k`
    godzina, uzupełniona spacją (0..23); to samo co `%_H`
  - `%l`
    godzina, uzupełniona spacją (1..12); to samo co `%_I`
  - `%m`
    miesiąc (01..12)
  - `%M`
    minuta (00..59)
  - `%n`
    znak nowego wiersza
  - `%N`
    nanosekundy (000000000..999999999)
  - `%p`
    lokalny odpowiednik `AM` lub `PM`; w wielu językach pusty
  - `%P`
    jak %p, lecz małymi literami

- `%r`
  czas w formacie 12-godzinnym (np. `11:11:04 PM`)
- `%R`
  godzina i minuty w formacie 24-godzinnym; to samo co `%H:%M`
- %s
  liczba sekund od 00:00:00, 1 stycznia 1970 UTC
- `%S`
  sekunda (00..60)
- `%t`
  tabulator
- `%T`
  czas; to samo co `%H:%M:%S`
- `%u`
  dzień tygodnia (1..7); 1 to poniedziałek
- `%U`
  numer tygodnia w roku, niedziela zaczyna tydzień (00..53)
- `%V`
  numer tygodnia ISO, poniedziałek zaczyna tydzień (01..53)
- `%w`
  dzień tygodnia (0..6); 0 oznacza niedzielę
- `%W`
  numer tygodnia w roku, poniedziałek zaczyna tydzień (00..53)
- `%x`
  lokalna reprezentacja daty (np. `19.03.2012`)
- `%X`
  lokalna reprezentacja czasu (np. `23:13:48`)
- `%y`
  dwie ostatnie cyfry roku (00..99)
- `%Y`
  rok

Domyślnie, `date` wypełnia pola numeryczne zerami. Następujące opcjonalne modyfikatory mogą być umieszczone po %:

- `-`
  (łącznik) nie wypełnia pola
- `_`
  (podkreślenie) wypełnianie spacjami
- `0`
  (zero) wypełnianie zerami
- `^`
  używa wielkich liter, jeśli to możliwe
- `#`
  zamienia wielkość liter, jeśli to możliwe

Polecenie sync

```
sync
```

Polecenie `sync` wymusza opróżnienie wszystkich buforów zapisu i przeniesienie ich zawartości do systemów plików oraz zaktualizowanie informacji pamiętanych w i-węzłach. Używane, gdy użytkownik

chce mieć pewność, że wszystkie zainicjowane wcześniej operacje zapisu (np. kierowane na urządzenia wymienne takie jak *pendrajwy*) zostały faktycznie zakończone.

Polecenie df

`df` *`[przełącznik...] [plik...]`*

Powoduje wyświetlenie informacji o dostępnej wolnej przestrzeni w systemie plików. Jeśli nie podano argumentu/argumentów *`plik...`*, prezentowana jest informacja o wszystkich systemach plików znanych systemowi operacyjnemu. Jeśli argument/argumenty podano, prezentowana jest informacja o systemie/systemach plików, na których rezydują pliki o podanych nazwach.

Na sposób prezentowania tej informacji mają wpływ następujące przełączniki:

- `--total`
  wyświetla podsumowanie całości
- `-h`
  wyświetla rozmiary w formacie czytelnym dla ludzi
- `-H`
  podobnie, ale używa potęg 1000 zamiast 1024
- `--no-sync`
  nie wywołuje sync przed pozyskaniem informacji (domyślnie)
- `--sync`
  wywołuje sync przed pozyskaniem informacji o użyciu

Polecenie du

`du` *`[przełącznik...] [plik...]`*

W pewnym sensie jest to przeciwieństwo polecenia `df`, jako że powoduje wyświetlenie informacji o przestrzeni **zajętej** przez pliki lub katalogi. W obliczeniach uwzględniany jest nie tylko rozmiar pliku, ale również fakt, że plik może zajmować na nośniku więcej miejsca niż, wymaga tego jego zawartość (dlaczego?). Na sposób prezentowania informacji mają wpływ następujące przełączniki:

- `-a`
  wypisuje podliczenia dla wszystkich plików, nie tylko katalogów
- `--apparent-size`
  wyświetla tzw. *rozmiar pozorny* zamiast faktycznego użycia dysku
- `-Bs`
  przelicza rozmiary według `s` przed ich wyświetleniem; dopuszczalne postaci `s` to:
  - liczba całkowita
  - liczba całkowita z przyrostkiem K, M, G, T, P, E, Z, Y (potęgi 1024) lub KB, MB, ... (potęgi 1000)
  - sam przyrostek
- `-b`
  równoważne `--apparent-size -B1`
- `-c`
  wyświetla podsumowanie całości
- `-D`
  rozwija tylko dowiązania symboliczne podane jako argumenty
- `-h`

wyświetla rozmiary w formacie czytelnym dla ludzi (np. `1K, 234M, 2G`)
- `--si`
  jak `-h`, lecz używa potęg 1000 zamiast 1024
- `-k`
  jak `-B1K`
- `-m`
  jak `-B1M`
- `-L`
  rozwija wszystkie dowiązania symboliczne
- `-P`
  nie podąża za żadnymi dowiązaniami symbolicznymi (domyślnie)
- `-S`
  nie uwzględnia rozmiarów podkatalogów

- `-s`
  wyświetla tylko podsumowanie dla każdego argumentu
- `--exclude=nazwa`
  pomija pliki pasujące do wzorca nazwy podanej w *nazwa*

Polecenie file

`file [`*`przełącznik...] plik...`*

To pożyteczne narzędzie istnieje w systemach klasy UNIX głównie z powodów doktrynalnych. Otóż filozofia tego systemu nigdy nie przewidywała, aby w nazwie pliku miała się znajdować jakakolwiek informacja o tym, co plik zawiera (co od zarania dziejów było standardem w systemach klasy Windows®). Rozwój Internetu zmienił to nastawienie i w chwili obecnej używanie identyfikujących plik przyrostków jest nie tylko na porządku dziennym, ale ponadto zostało ujęte w normach międzynarodowych, jednak zawartość pewnych plików nadal może być zagadką. Na przykład pliki, które oznaczone są jako '`x`' (wykonywalne) mogą mieć bardzo różną zawartość: mogą być programami skompilowanymi, nazywanymi w slangu „*binarkami*”, ale mogą być też zwykłymi plikami tekstowymi, zawierającymi skrypty w przeróżnych językach interpretowanych (bash, awk, tcl, Python, Perl, itd.). Można oczywiście próbować oglądać takie pliki od środka i wydawać osąd na podstawie wzrokowej analizy ich zawartości, ale nie zawsze jest to wygodne. Ponadto, może się również zdarzyć, że na skutek nieszczęśliwego wypadku plik może stracić nazwę i warto posiadać narzędzie, które samo przeanalizuje jego zawartość i postawi hipotezę odnośnie tego, czym ten plik jest w istocie.

Polecenie `file` wykonuje tę czynność na postawie specjalnej bazy danych nazywanej „*magic numbers*”. Jest to zbiór prostych heurystyk pozwalających sprawdzić, czy dany plik ma (lub nie) pewne ewidentne cechy (np. zbiór użytych znaków, obecność charakterystyczny sygnatur, itp.).

W najprostszym przypadku wystarczy wskazanie nazwy pliku, a `file` odpowie stosowną informacją, np.:

```
$ file win/explorer.exe
win/explorer.exe: PE32+ executable (GUI) x86-64, for MS Windows
```

Polecenie sleep

`sleep `*`liczba[przyrostek]`*

Polecenie sleep wywołuje efekt bezczynności przez czas określany argumentem. Przydatne głównie w skryptach, ale może być użyteczne także w pracy konsolowej np. w celu wykonania pewnego polecenia dopiero po upływie pewnego czasu. Dopuszczalne przyrostki to:

- `s`
  sekundy (domyślne)
- `m`
  minuty
- `h`
  godziny
- `d`
  dni

Polecenie diff

`diff` *`[przełącznik...] plik1 plik2`*

Polecenie `diff` służy do porównania zawartości wskazanych plików – w najprostszym przypadku porównuje zawartość dwóch plików, `plik1` i `plik2`. Jeśli `plik1` jest katalogiem, a `plik2` nie, `diff` porównuje plik z katalogu `plik1`, którego nazwa jest taka sama, jak `plik2`, i odwrotnie. Jeśli `plik1` i `plik2` są katalogami, `diff` porównuje pliki o zgodnych nazwach, istniejące w obu katalogach, w kolejności alfabetycznej. To porównanie nie jest ponawiane w podkatalogach, chyba że podano opcję `-r`. Jeśli porównywane pliki nie różnią się, `diff` nie odzywa się ani słowem.

Opcje `diff` na ogół zaczynają się od `-`, więc zwykle nazwy plików `plik1` oraz `plik2` nie mogą zaczynać się od `-`, jednak argument `--` traktuje pozostałe argumenty jako nazwy plików, nawet jeśli zaczynają się one od `-`. Konwencja ta jest stosowana również przez wiele innych narzędzi.

Informacje wyprowadzanie przez `diff` mogą być bezpośrednio użyte przez polecenie `patch`. Zapoznanie się z możliwościami tego narzędzia pozostawia się słuchaczowi.

Należy podkreślić, że `diff` został stworzony do wyszukiwania różnic w plikach tekstowych, a postać dostarczanych przez niego informacji ułatwia np. identyfikowanie miejsc, w których modyfikowano kod źródłowy. Z tego też powodu `diff` nie nadaje się do porównywania plików binarnych (np. plików graficznych). Do tego celu służy polecenie `cmp`.

Najczęściej używane przełączniki `diff` to:

- `-i`
  ignorowanie zmian w wielkości liter, duże i małe litery są uznawane za równoważne
- `-w`
  ignorowanie wszystkich odstępów przy porównywaniu plików
- `-b`
  ignorowanie zmian w ilości odstępów
- `-B`
  ignorowanie zmian, które jedynie dodają lub usuwają puste linie
- `-q`
  poinformowanie jedynie o tym, czy pliki się różnią, bez podawania szczegółów na temat różnic
- `-r`
  rekurencyjne porównanie wszystkich podkatalogów, jeśli porównywane są katalogi
- `-N`
  jeśli podczas porównywania katalogów plik istnieje jedynie w jednym z katalogów, będzie traktowany tak, jakby był obecny w drugim katalogu, ale pusty
- `-P`

jeśli podczas porównywania katalogów plik istnieje jedynie w drugim katalogu, będzie traktowany tak, jakby był obecny w pierwszym katalogu, ale pusty

- `-s`
  poinformowanie, jeśli oba pliki są identyczne
- `-x wzorzec`
  podczas porównywania katalogów, ignorowanie plików i podkatalogów, których nazwy pasują do wzorca `wzorzec`

Polecenie cmp

`cmp` *`[przełącznik...] plik1 plik2`*

`cmp` porównuje dwa pliki dowolnego typu i wypisuje wynik tego porównania (albo nie). Domyślnie `cmp` zachowuje pełne godności milczenie, gdy pliki są identyczne. W przeciwnym przypadku `cmp` zgłasza numer bajtu i linii, w których wykryto pierwszą różnicą. Bajty i linie są numerowane od jeden.

Najprzydatniejsze przełączniki to:

- `-c`
  wypisuje różniące się znaki
- `-i ile`
  ignoruje wszelkie różnice w *ile* początkowych bajtach każdego z plików. Traktuje pliki zawierające mniej niż *ile* bajtów tak, jakby były puste
- `-l`
  dla każdej różnicy wypisuje numer bajtu (dziesiętnie) i wartości różniących się bajtów (ósemkowo).

Zadania do wykonania w czasie zajęć

1. Czy kolejność zestawów *kto-jak-co* wymienionych w jednym poleceniu `chmod` ma znaczenie? Podaj przykład, który uzasadnia twoją odpowiedź.
2. Oznacz jako '`x`' plik, który nie jest plikiem wykonywalnym i *wykonaj* go. Jak zareagował system operacyjny?
3. Zapoznaj się z zawartością pliku `/etc/passwd`. Czy są tam użytkownicy, który używają shella innego niż `/bin/bash`?
4. Spróbuj zapoznać się z zawartością pliku `/etc/shadow`. Co się stało i dlaczego?
5. Zapoznaj się z zawartością pliku /`etc/group`. Czy są tam grupy, do których należy więcej niż jeden użytkownik?
6. Odszukaj w systemie operacyjnym plik wykonywalny `busybox` i stwórz do niego link symboliczny o nazwie `cp`. Uruchom link symboliczny (upewnij się, że uruchamiasz *busyboksa*, a nie oryginalne polecenie `cp`). Sprawdź, czy możesz po swoim dowiązaniu oczekiwać zachowania zgodnego z `cp`. Jakie jeszcze polecenia potrafi emulować `busybox`?
7. Czy przy pomocy polecenia `ln` można stworzyć zapętloną strukturę katalogów? Jeśli tak, to jak się po niej poruszać? Jak rozerwać taką pętlę?
8. Stwórz plik, a następnie dowiązanie symboliczne, które prowadzi do niego. Usuń plik. Jak teraz zachowuje się dowiązanie?
9. Stwórz plik, a następnie dowiązanie do niego, a następnie dowiązanie do tego dowiązania. Zbadaj właściwości tego tworu. Spróbuj rozbudować łańcuch dowiązań.
10. Użyj polecenia `chmod` w celu zmiany praw dowiązania symbolicznego. Co się stało?

11. W systemie operacyjnym istnieje para programów nazywających się `true` i `false`. Zbadaj ich działanie. Jakiego skutku można oczekiwać, gdy w pliku `/etc/passwd` pewien użytkownik będzie miał jako swoją powłokę wpisany jeden z tych te programów? Czy obecnie w naszym systemie jest taki użytkownik?
12. Przeanalizuj sposób, w jaki program `diff` prezentuje odnalezione różnice między plikami. Zaplanuj i przeprowadź stosowny eksperyment, posługując się kolejno plikami identycznymi, różniącymi się nieznaczne, różniącymi się znacznie i kompletnie innymi Upewnij się, że potrafisz przewidzieć, co powie ci `diff`, jeśli znasz dobrze zawartość obu plików.
13. Jeśli czasy w metadanych pliku byłyby reprezentowane przy pomocy liczb 32-bitowych, to kiedy skończyłaby się ich pojemność?
14. Czy polecenie `file` jest w stanie odróżnić program w *C* od programu w *C++*? Przeprowadź eksperyment.
15. Posługując się poleceniami `watch` i `date` skonstruuj prosty, odświeżany co sekundę zegarek, który prezentuje informację o czasie w sposób przedstawiony poniżej (ostatnia linia prezentuje sekundy):

    ```
    22 październik
    2014
    13:44
    00
    ```

16. Użyj polecenia `date` do wypisania daty w formie zgodnej z normą ISO 8601, czyli np. `2014-10-30`
17. Który tydzień roku mamy obecnie?
18. Co to jest „*rozmiar pozorny*” prezentowany przez polecenie du?
19. Co pojawi się na ekranie, jeśli użyjesz `du` z przełącznikiem `-B512`?
20. Sprawdź, ile przestrzeni dyskowej zajmuj twój katalog domowy.
21. Wydaj z konsoli polecenie, które powie ci, kiedy upłynęła minuta.