Wprowadzenie do systemu UNIX
Operacje w systemie plików

Wprowadzenie

Na poprzednich zajęciach poznaliśmy podstawowe polecenia pozwalające na wyszukiwanie wzorców w plikach tekstowych. Teraz nadszedł czas na przedstawienie metod poruszania się w systemie plików.

W dalszej części laboratorium omówione zostaną metody wyszukiwania plików, zarządzania uprawnieniami i dostępną przestrzenią, dowiązania twarde i symboliczne oraz systemy kompresji danych.

System plików (ang. *filesystem*) to podsystem jądra systemu służący do zarządzania plikami, przydzielania miejsca na ich składowanie, administrowania uprawnieniami dostępu do nich oraz udostępniania plików użytkownikom.

Podstawowym narzędziem służącym do wyszukiwania plików na podstawie nazwy lub innych atrybutów jest polecenie **find**. Dzięki niemu możliwe jest wyszukiwanie plików z określoną (także jako przedział) datą dostępu lub modyfikacji, określonego właściciela indywidualnego lub grupowego, wskazanego typu lub liczby istniejących struktur *i-node* na niego wskazujących.

Dla znalezionych plików możliwe jest wykonanie określonych akcji, jak usunięcie pliku. Istnieje także możliwość wywołania wcześniej określonej komendy.

Zapoznaj się ze składnią i możliwościami polecenia *find*. Potem odpowiedz na pytania:

- Jakie są możliwe akcje dla znalezionych plików i w jaki sposób są one tworzone?
- Jak wyszukiwać pliki na podstawie przyznanych użytkownikom uprawnień do nich?
- W jaki sposób wyznaczane są czasy (np. modyfikacji lub dostępu) do plików?
- Jak traktowane są dowiązania symboliczne (ang. *symbolic links*) podczas przeszukiwania drzewa katalogów?

Ważnym zadaniem systemu plików jest zarządzanie uprawnieniami użytkowników względem plików i katalogów obecnych w systemie. Na początek przedstawmy spotykane typy plików (w nawiasie podane zostały oznaczenia literowe danych typów):

- plik regularny (-) – zawiera dane,
- katalog (ang. *directory*) (d) – zawiera inne pliki i katalogi,
- urządzenie blokowe (b) – dysk, partycja dyskowa (buforowany dostęp do urządzeń poprzez warstwę abstrakcji ukrywającą ich budowę),
- urządzenie znakowe (ang. *character device*) (c) – konsole i terminale, porty szeregowe itd. (bezpośredni dostęp do urządzenia),
- dowiązanie symboliczne (ang. *symbolic link*) (l) – „skrót” do pliku/katalogu,

- gniazdo uniksowe (ang. *socket*) (s) – plik specjalny pozwalający na komunikację pomiędzy lokalnie działającymi procesami, podobny do komunikacji sieciowej,
- potok nazwany (ang. *named pipe*) (p) – plik specjalny służący do przekazywania danych pomiędzy lokalnymi procesami, przypominający kolejkę (dane odczytywane są w kolejności, w jakiej zostały zapisane).

Typ pliku można sprawdzić, wyświetlając go w formie długiej przy pomocy polecenia `ls -l`.

Każdy plik ma przypisane do niego uprawnienia. Jest to realizacja modelu dostępu DAC (ang. *Discretionary Access Control*), w którym o możliwych akcjach względem pliku decyduje jego właściciel lub administrator systemu (konto *root*). Do pliku w systemie przypisane są uprawnienia właściwe dla jego właściciela indywidualnego (ang. *user*), właściciela grupowego (ang. *group*) oraz pozostałych użytkowników (ang. *others*).

Istnieją trzy podstawowe typy uprawnień:

- odczyt (ang. *read*) – odczytanie zawartości pliku lub zawartości (listy plików) katalogu,
- zapis (ang. *write*) – zmianę zawartości pliku lub katalogu (dodawanie lub usuwanie plików w nim obecnych),
- wykonanie (ang. *execute*) – użycia pliku jako skryptu powłoki lub programu, a w przypadku katalogu - użycie go jako katalogu bieżącego.

Uprawnienia można nadawać osobno dla każdej z grup: właściciela indywidualnego, właściciela grupowego oraz pozostałych użytkowników. Opisuje się je zazwyczaj przy pomocy wartości liczbowych lub ciągu dziewięciu znaków, po trzy dla każdej grupy, gdzie: **r** – odczyt, **w** – zapis, **x** – wykonanie.

Przykładowo, ciąg:

```
rw-r-x---
```

odpowiada nadanym uprawnieniom odczytu i zapisu (bez uprawnienia wykonywania) dla właściciela indywidualnego, odczytu i wykonania (bez uprawnienia zapisu) dla właściciela grupowego i odmowie jakichkolwiek uprawnień dla wszystkich pozostałych użytkowników.

Każdą z grup uprawnień można opisać wartością liczbową, gdzie:

- 4 – uprawnienie *read*,
- 2 – uprawnienie *write*,
- 1 – uprawnienie *execute*.

Wartości te sumowane są dla poszczególnych grup, zatem powyższy przykład można zapisać ciągiem **650**.

Liczby te biorą się z zapisu binarnego uprawnień. Jeżeli wyobrazimy sobie nasze uprawnienia jako maskę bitową, gdzie o oznacza brak danego uprawnienia, a 1 – jego nadanie, to dostaniemy sytuację, jak w Tabela 1.

Stąd też, np. uprawnienie `r-x` to w zapisie binarnym 101, a więc w zapisie dziesiętnym – 5.

Pozycja	2	1	0
Uprawnienie	r	w	x
Wartość dziesiętna	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

Tabela 1: Wartości poszczególnych uprawnień.

Rozszerzeniem podstawowego modelu uprawnień są trzy dodatkowe opcje:

- SUID (ang. *Set User ID upon execution*) - program zawarty w uruchamianym pliku wykonywany jest z uprawnieniami właściciela indywidualnego pliku, a nie użytkownika uruchamiającego program
- SGID (ang. *Set Group ID upon execution*):
  - w przypadku plików – program zawarty w uruchamianym pliku wykonywany jest z uprawnieniami właściciela grupowego pliku, a nie użytkownika uruchamiającego program,
  - w przypadku katalogów – pliki dziedziczą po katalogu, w którym się znajdują, właściciela grupowego,
- *Sticky bit*:
  - w przypadku plików wykonywalnych – program pozostaje w pamięci nawet po jego wyłączeniu, co przyspiesza jego ponowne włączenie (w praktyce rozwiązanie to nie jest już stosowane i przez większość systemów ustawienie *sticky bit* na pliku jest ignorowane),
  - w przypadku katalogów – pliki znajdujące się w tym katalogu może usunąć tylko właściciel tych plików (normalnie jest to posiadacz prawa zapisu do katalogu).

Przykładem zastosowania SUID jest wykorzystywanie poleceń **passwd** oraz programów inicjujących komunikację sieciową, jak **ping** lub **traceroute**. Do ich działania wymagane są uprawnienia administracyjne, których zazwyczaj nie posiada uruchamiający je użytkownik.

Zadania

1. Czy znajdując się w katalogu `$HOME`, jednym wywołaniem komendy **ls** można obejrzeć zawartość wszystkich katalogów do samego końca drzewa plików (czyli w sposób rekursywny)? Sprawdź to w manualu.
2. Ustal, w jaki sposób oznaczane są ustawione uprawnienia dodatkowe (SUID, SGID i SVTX).
3. W katalogu `$HOME/c2/text` utwórz plik zawierający strony podręcznika dla komendy **ls**. Sprawdź prawa dostępu utworzonego pliku. W jaki sposób i przez kogo mogą zostać one zmienione?

   Podpowiedź: sprawdź pomoc poleceń **chown**, **chgrp** oraz **chmod**.
4. Zmień prawa dostępu do utworzonego pliku tak, aby tylko jego właściciel indywidualny mógł go odczytać i zapisać. Ustaw prawa dostępu pliku i katalogu, w którym plik się znajduje tak, aby pliku nie można było usunąć komendą **rm**.
5. Skopiuj plik na dowolny inny. Jakie są prawa dostępu nowego pliku? Zmień prawa dostępu nowego pliku tak, aby tylko właściciel indywidualny i inni członkowie jego grupy podstawowej mogli go odczytać.
6. W jaki sposób ustawiane są domyślne prawa do tworzonych przez użytkownika plików? Jak to zmienić?

   Podpowiedź: polecenie **umask**.

7. Podaj nazwę pliku w katalogu domowym (rekursywnie) o największym rozmiarze. Jaki jest największy plik tylko w katalogu domowym i jego bezpośrednich podkatalogach?

8. Podaj nazwę pliku w katalogu domowym (rekursywnie) o największej ilości bloków.

9. Korzystając z komendy **find**, wyszukaj i wylistuj w postaci długiej wszystkie pliki regularne, które znajdują się w katalogu `/usr/include` i jego podkatalogach, a ich nazwa zaczyna się na literę **c**.

10. Czy w Twoim katalogu domowym (z podkatalogami) są pliki modyfikowane wcześniej niż 2 dni temu? Jak to sprawdzić? Wylistuj je w postaci długiej. Zrób to raz z uwzględnieniem podkatalogów, a raz – bez.

11. Znajdź liczbę plików regularnych w `/usr/include` (bez podkatalogów) które zaczynają się na **m** i kończą na **.h** oraz których rozmiar nie przekracza 12KB.

12. Przy pomocy komendy **tar** stwórz archiwum katalogu **c2**, a następnie dokonaj jego kompresji (np. przy pomocy programu **gzip**). Powtórz tę operację w jednym kroku z użyciem dwóch komend, a następnie jednej.

    Następnie usuń katalog **c2**, a następnie odtwórz go ze zbioru ***.tar.gz** (lub ***.tgz**).

13. Sprawdź, ile miejsca na dysku zajmuje twój katalog osobisty i jakie ograniczenia zostały nałożone na jego rozmiar (komenda **quota**).

14. Zapoznaj się ze stronami manuala dla komend **df** i **free**. Określ rozmiary systemów plików dostępnych w systemie oraz stopień ich zajętości w odniesieniu do liczby bloków danych i liczby struktur *i-node*.

15. Zidentyfikuj punkty montowania oraz typy używanych systemów plików za pomocą komend **mount** i **lsblk**. Jak uzyskać numery seryjne fizycznych dysków?

    Podpowiedź: użyj odpowiednich opcji polecenia **lsblk**. W przypadku ich braku (głównie na starszych systemach), możesz wykorzystać polecenie **udevadm**.

16. Korzystając z komend **du** oraz **sort**, zaproponuj takie ich powiązanie, które pozwoli wylistować w kolejności malejącej listę katalogów z katalogu `/usr` uporządkowaną według wykorzystania przestrzeni dyskowej przez te katalogi i ich podkatalogi.

    Jeśli nie istnieją pliki wskazywane w treści polecenia (a powinny), to należy je utworzyć zgodnie z instrukcją z poprzednich zajęć.

17. W katalogu `$HOME/c2/text`, korzystając z polecenia **ln** utwórz dowiązanie symboliczne o nazwie **symbol** do pliku `$HOME/c2/bin/moj_ls`, zaś w katalogu `$HOME/c2/bin` dowiązanie twarde o nazwie **twardy** do pliku `$HOME/c2/text/ls.txt`. Przejdź do katalogu `$HOME/c2` i wylistuj jego zawartość rekurencyjnie w postaci długiej z numerami *i-node*. Jakie widzisz zależności w numerach *i-node*, rozmiarach plików i ich nazwach? Usuń plik `$HOME/c2/text/ls.txt` i dokonaj listowania ponownie. Jakie zależności występują obecnie? Usuń plik `$HOME/c2/bin/mój_ls` i wykonaj listowanie. Co stało się z dowiązaniem twardym?

Zadanie sprawdzające

Poniższe zadanie wykorzystuje kompleksowo wiedzę z tego laboratorium. Postaraj się wykonać je samodzielnie w domu. Jeżeli masz z nim problemy, przestudiuj ponownie materiały źródłowe, rozwiąż wcześniejsze zadania i podejmij kolejną próbę.

Podaj pełną komendę zwracającą konkretną wartość (nie należy np. liczyć wierszy „ręcznie"):

1. Znajdź plik w swoim katalogu domowym, który był najdawniej modyfikowany.
2. Ile katalogów znajduje się w `/usr/include`?
3. Ile jest plików w `/usr/include` (bez podkatalogów), których nazwa składa się z 6 liter, zaczyna się na **m** i kończy na **.h**?
4. Ile jest plików regularnych w katalogu i podkatalogach `/usr/include`, które mają rozmiar większy niż 12000B.
5. Ile jest plików w `/usr/bin` o wielkości do 1MB?

Podsumowanie komend

Na zajęciach przedstawione zostały następujące komendy:

Grupa	Komenda
Wyszukiwanie plików	find, grep
Rozmiar plików	du
Zajętość miejsca i pamięci	df, free
Wyświetlanie	stat, ls, lsblk
Kompresja	tar, zip, unzip, gzip
Dowiązania	ln
Przydziały miejsca	quota, quotacheck
Montowanie	mount
Uprawnienia	chmod, chown, chgrp, umask

Przydatne polecenie: xargs

Polecenie **xargs** pozwala na pobranie teksu (ciągów znakowych) ze standardowego wyjścia jednej komendy i wykorzystanie ich jako argumenty wskazanego polecenia.

Przykładowo, zamiast komendy:

```
find . -exec touch '{}' \;
```

Można wykorzystać:

```
find . -print | xargs touch
```

Jest to oczywiście dosyć prosty przykład wykorzystania komendy **xargs** (pozwala na modyfikację czasu dostępu wszystkich plików w bieżącym katalogu), którego funkcjonalność można w prosty sposób osiągnąć bez stosowania go, jednak pozwala na zilustrowanie możliwych zastosowań.