Politechnika Wrocławska

Systemy operacyjne 2

Laboratorium nr 4

Program `find`

Szymon Datko

szymon.datko@pwr.edu.pl

Wydział Informatyki i Telekomunikacji,

Politechnika Wrocławska

semestr letni 2021/2022

HR EXCELLENCE IN RESEARCH

W skrócie – `find .`

- Narzędzie ułatwia nawigowanie po drzewie katalogów.
- Zadania podobne, jak do tej pory, ale na większą skalę.
- Nie będziemy ograniczać się do pojedynczych poziomów podkatalogów.

(punkt widzenia: administrator szukający plików)

Źródło obrazka: https://knowyourmeme.com/memes/i-will-find-you-and-i-will-kill-you

Politechnika Wrocławska

Część I

Omówienie zagadnień

Politechnika Wrocławska

Z czego wynika dostęp do pliku?

- Każdy plik w systemie ma określonego właściciela: użytkownika i grupę.
- Podstawowy mechanizm kontroli opiera się na 3 poziomach uprawnień.
  - Określa się osobne uprawnienia w dostępie do pliku dla:
    - użytkownika (właściciela) – `user` – `u`,
    - grupy (właścicieli) – `group` – `g`,
    - pozostałych osób – `other` – `o`.
  - Niektóre systemy plików wspierają także rozszerzoną liczbę poziomów.
  - Dla dociekliwych: `man setfactl`.
- Prawa dostępu dzielą się na 3 różne akcje, zapisane jako liczby ósemkowe:
  - wykonanie – *execute* – `x` – wartość `1`,
  - zapis – *write* – `w` – wartość `2`,
  - odczyt – *read* – `r` – wartość `4`,
  - np. `754` oznacza uprawnienia `rwx r-x r--`, czyli `u=rwx g=rx o=r`.
- Właściwy dostęp (możliwość wykonania akcji) może wynikać z kilku źródeł.
  - Bycia właścicielem, członkiem grupy, albo z ogólnych ustawień pliku, itd.

Program `find`

- Narzędzie do odnajdywania plików w drzewie katalogów.
- Działa na zasadzie testowania plików względem podanego **wyrażenia**.
- Pozwala wykonać dodatkowe akcje na znalezionych (pasujących) plikach.
- **Uwaga!** Program ten stosuje inną konwencję opcji niż typowa w powłoce!

```
NAME
       find - search for files in a directory hierarchy

SYNOPSIS
       find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...] [expression]

DESCRIPTION
       This  manual page documents the GNU version of find.  GNU find searches the directory tree
       rooted at each given starting-point by evaluating the given expression from left to right,
       according  to  the rules of precedence (see section OPERATORS), until the outcome is known
       (the left hand side is false for and operations, true for or), at which point  find  moves
       on to the next file name.  If no starting-point is specified, `.' is assumed.

       If  you  are  using find in an environment where security is important (for example if you
       are using it to search directories that are writable by other users), you should read  the
       `Security  Considerations' chapter of the findutils documentation, which is called Finding
       Files and comes with findutils.  That document also includes a lot more detail and discus-
       sion than this manual page, so you may find it a more useful source of information.
```

Program `find` – punkt startowy

- Określa ścieżkę, od której należy rozpocząć poszukiwania.
  - Standardowo jest to też pierwszy element testowany wyrażeniem.
- Domyślnie jest to pierwszy parametr programu niebędący opcją.
  - Czyli pierwszy argument, który nie rozpoczyna się od znaku minusa (`-`).
  - `find -L katalog/ -iname '*.txt'`
    `^^^^^^^^`
  - Pytanie dla dociekliwych: jak wejść do katalogu o nazwie `-` (minus)?
- Jeśli jest to katalog – zostaną przetestowane **wszystkie** jego pliki.
  - Domyślnie zostaną uwzględnione także podkatalogi i **cała** ich zawartość.
  - Opcje `-mindepth` i `-maxdepth` pozwalają ograniczyć zakres podkatalogów.
- Można podać więcej, niż jeden punkt startowy.
- Przykłady użycia:
  - `find katalog/`
  - `find katalog1/ katalog2/ katalog3/`

Program `find` – wyrażenia (1/3)

- Sprawdzenie uprawnień z perspektywy bieżącego użytkownika.
  - `-executable` – możliwość wykonania pliku.
  - `-readable` – możliwość przeczytania zawartości pliku.
  - `-writable` – możliwość zapisania/nadpisania pliku.
- Testowanie nazwy pliku.
  - `-name wzór` lub `-iname wzór` (drugi wariant ignoruje wielkość liter).
  - We wzorze mogą pojawić zwykłe ciągi, lub znaki specjalne `*`, `?`, itp.
  - Testowana jest zawsze sama nazwa pliku (*basename*), czyli ciąg bez `/`.
  - Zastosowanie wzorca `a/b`, czyli `-name a/b`, nigdy nic nie odnajdzie!
- Sprawdzanie ścieżki do pliku – operator `-path wzór`.
  - Działa analogicznie do testowania nazwy, ale na pełnej ścieżce (obecne `/`).
- Kontrola typu pliku – operator `-type typ`.
  - Typ jest określany literą: `d` (katalog), `f` (plik zwykły), `l` (dowiązanie), itp.
  - W przypadku dowiązań warto pamiętać jeszcze o podobnej opcji `-xtype`.

Program `find` – wyrażenia (2/3)

- Testowanie rozmiarów plików – operator `-size N[jednostka]`.
  - Pozwala sprawdzić czy plik zajmuje $N \in \mathbb{Z}$ jednostek w systemie plików.
  - Jednostkę określa opcjonalna pojedyncza litera, domyślnie jest to blok (`b`).
  - Przydatne jednostki: `c` (Bajt), `k` (kibibajt), `M` (mebibajt), itp.
  - Poprzedzając `N` znakiem `+`/`-` szukamy plików większych/mniejszych od `N`.
  - **Uwaga!** Rozmiar plików jest zaokrąglany w górę w ramach jednostki!
  - W szczególności: `-size -1M` to nie to samo co `-size -1048576c` (**!!!**).
- Sprawdzanie czy plik/katalog jest pusty – opcja `-empty`.
  - Czyli właściwie to samo co `-size 0`.
- Porównanie numeru i-węzła z plikiem – opcja `-samefile ścieżka`.
  - Przydatne przy poszukiwaniu dowiązań twardych do jakiegoś pliku.
- Sprawdzenie ścieżki w dowiązaniu – operator `-lname ścieżka`.
  - W przypadku dowiązań miękkich: porównywana jest wskazywana ścieżka.
- Kontrola właścicielstwa pliku – operatory `-user nazwa` i `-group nazwa`.

Program `find` – wyrażenia (3/3)

- Operatory związane z kontrolą czasów dostępu/utworzenia/modyfikacji:
  - dla większości argument stanowi liczba całkowita `N` $\in \mathbb{Z}$, określająca czas:
    - w minutach (`-amin`, `-cmin`, `-mmin`),
    - w dniach (`-atime`, `-ctime`, `-mtime`),
  - argument `N` można poprzedzić znakiem `+`/`-`, podobnie jak w `-size`,
    - wyszukamy wtedy pliki modyfikowane więcej/mniej niż `N` dni temu,
    - tutaj również wielkości są zaokrąglane i należy o tym pamiętać,
  - istnieją także opcje pozwalające określić czas względem jakiegoś pliku,
    - to między innymi `-anewer`, `-cnewer` i `-newer` (argument: ścieżka),
    - nie ma operatora przeciwnego, należy użyć negacji w razie potrzeby.
- Sprawdzanie bitów, kontrolujących uprawnienia dostępu do plików:
  - w tym celu wykorzystujemy operator `-perm maska`,
  - argument może mieć formę liczbową (np. `020`) albo symboli (np. `g=w`),
  - `maska` może być poprzedzona znakami `-` lub `/`, zmieniającymi działanie,
    - `maska` (bez niczego) wymaga dokładnego dopasowania uprawnień,
    - `-maska` weryfikuje ustawienie wszystkich podanych bitów uprawnień,
    - `/maska` sprawdza czy ustawiony jest dowolny z podanych bitów.

Program `find` – łączenie wyrażeń

- Wyrażenia są wykonywane od lewej do prawej strony.
  - Jeśli któreś po drodze zakończy się fałszem, wszystkie kolejne są pomijane.
  - Wynika to z faktu, że...
- Domyślnie implikowana jest koniunkcja logiczna (AND).
  - Można określić ją jawnie przez operator `-a` lub `-and`.
  - Przykład – poniższe wyrażenia są zasadniczo równoważne:

```
find . -type f -and -iname '*.jpg' -print
find . -type f -a -iname '*.jpg' -print
find . -type f -iname '*.jpg' -print
```

- Alternatywa logiczna (OR) – wykonanie następuje do pierwszej prawdy.
  - Realizowana jest przez operator `-o` lub `-or`.
- Negacja logiczna – pozwala odwrócić wynik wyrażenia.
  - Stosuje się operator `!` lub `-not`.
- Grupowanie wyrażeń jest możliwe przez operatory nawiasów `\(` i `\)`.
  - Ukośnik `\` zapobiega interpretowaniu nawiasów przez powłokę.
- Operatory `-not` `-and` i `-or` stanowią nie-POSIXowe rozszerzenia.

Program `find` – akcje

- Typowo stanowią zakończenie jakiegoś (pod)wyrażenia.
- `-print` – wyświetlenie ścieżki do znalezionego pliku.
  - Jest to domyślnie wykonywana akcja, kiedy nie podano żadnej innej.
- `-printf format` – rozbudowany, konfigurowalny wariant wyświetlenia.
  - Umożliwia wyświetlenie na przykład samej nazwy pliku czy rozmiaru.
  - Pola w formacie poprzedzone są znakiem %, np. `%h`, `%p`, `%s`.
- `-ls` – zwrócenie informacji jak program `ls` z opcjami `-d`, `-i`, `-l` oraz `-s`.
- `-prune` – przerwanie przeszukiwania bez wchodzenia do katalogu.
  - Zazwyczaj łączone z flagą `-o` (OR) do odfiltrowania części wyników.
  - Przykład – wyszukanie plików, które nie należą do katalogu `.git`:
    ```
    find . -name '.git' -prune -o -type f -print
    ```
- `-delete` – usunięcie pliku, który został dopasowany do wyrażenia.
- `-exec komenda {} \;` – wykonanie komendy na każdym pasującym pliku.
  - Fragment `{}` zastępuje ścieżkę do pliku, a `\;` oznacza koniec komendy.
  - Bez ukośnika `\` znak średnika `;` zostałby zinterpretowany przez powłokę.
  - Znak `+` może zastąpić `;` i komenda wykona się raz dla wszystkich plików.

Zliczenie znalezionych plików

- W programie `find` nie ma dedykowanej akcji do tego celu.
- Trzeba samodzielnie przetworzyć wynik działania programu `find`.
- Można w tym celu skorzystać z pętli i użyć operatorów matematycznych.
  - Przy pętli `for` należy uważać na możliwość wystąpienia spacji w ścieżkach.
  - Pomocne może być użycie akcji `-printf` i zwracać jakiś stały ciąg znaków.
  - Na przykład tak: `find katalog wzorzec -printf '.\n'`.
- Lepszym pomysłem może być zastosowanie programu `wc` do zliczenia linii.
  - Wystarczy dopisać `| wc -l` na końcu komendy z programem `find`.
    - Przed – zwrócenie listy plików: `find katalog wzorzec`.
    - Po – policzenie plików na liście: `find katalog wzorzec | wc -l`.
  - Wykorzystany jest tu tak zwany mechanizm potoków.
  - Dokładniej działanie tego mechanizmu omówimy na kolejnych zajęciach ;-)
    - Na potrzeby bieżących zajęć można przyjąć do wiadomości podany przykład i po prostu stosować go w rozwiązaniach w tej postaci.

Z czego wynika dostęp do pliku? – uzupełnienie

- W zadaniach dot. uprawnień proszę zwrócić uwagę czego one dotyczą.
  - W szczególności należy po prostu zadać sobie jedno bardzo ważne pytanie:
  - czy chodzi o możliwość wykonania **akcji**, czy o ustawienie **bitów** dostępu?
- Sprawdzenie ustawienia bitów dostępu realizujemy operatorem `-perm`.
- Skontrolowanie czy możemy wykonać jakąś akcję może być nietrywialne.
  - Samo jedno sprawdzenie np. `-perm -u=x` może nie być wystarczające.
  - W takich wypadkach lepiej posłużyć się posłużyć dedykowanymi opcjami.
  - Czyli np. zastosować `-readable` zamiast szeregu wymyślnych `-perm`.
- Co oznaczają bity dostępu w kontekście katalogu:
  - wykonanie – `x` – możliwość wejścia do niego i dostępu do jego plików,
    - konieczne, aby móc przejść/uwzględnić katalog w ścieżkach i otwierać same zawarte pliki,
  - zapis – `w` – prawo do zmiany jego zawartości (tworzenie i usuwanie plików),
    - zasadniczo bezużyteczne bez prawa wykonania (dostępu do plików w katalogu),
  - odczyt – `r` – pozwala przeczytać listę plików obecnych w danym katalogu,
    - daje możliwość wyszukania plików na dysku (tylko w danym katalogu, bez podkatalogów!),
  - więcej informacji: https://www.hackinglinuxexposed.com/articles/20030424.html.

Politechnika Wrocławska

Część dla dociekliwych

Zadanie dodatkowe

Politechnika Wrocławska

Zadanie domowe

`glob` - z czym to się je?

W powłoce Bash możemy definiować ścieżki przy pomocy pewnych wzorców, przypominających wyrażenia regularne - na przykład `./plik*.sh`. Proszę zapoznać się z tym zagadnieniem, omówić jakie specjalne znaki/sekwencje pozwalają na wymuszenie określonego dopasowania ścieżek oraz jak ich stosowanie ma się do zwykłych napisów (ciągów pomiędzy apostrofami lub cudzysłowami).

W szczególności proszę opisać czy możemy jakoś sensownie zdefiniować wzorzec do ukrytych plików z rozszerzeniem txt, znajdujących się w katalogu, zawierającym spacje w nazwie. A co się stanie, jeśli w nazwie jakiegoś pliku pojawi się znak *, znak / lub znak nowej linii? (Czy w ogóle jest to możliwe?)