Kurs administrowania systemem Linux 2025

Lista zadań na pracownię nr 3

Na zajęcia 12 i 13 marca 2025

Zadanie 1 (1 pkt). U pewnego melomana-bałaganiarza pliki MP3 są zapisane w różnych katalogach, pod przypadkowymi nazwami i przemieszane z plikami innych typów, np.

```
$ ls -R
.:
a.jpg                    O2EBAioCURP2Z6ly.mp3  other/  veUZm4VhkOXbBbWi.mp3
Vn8K\ tMC\ fgeEL8M.mp3  YPB1mEyrye5J93np.mp3

./other:
cxNLk31byciZHESj.mp3  README.txt
```

Na szczęście tagi ID3 w tych plikach są zapisane poprawnie, więc za pomocą polecenia `mp3info(1)` możemy odczytać wykonawcę i tytuł utworu, np.:

```
$ mp3info Vn8K\ tMC\ fgeEL8M.mp3
File: Vn8K tMC fgeEL8M.mp3
Title:   Act I. Finale                  Track: 10
Artist:  Pyotr Ilyich Tchaikovsky
Album:   Swan Lake                      Year:  1998
Comment:                                Genre: Classical [32]
```

Napisz skrypt, który uruchomiony w danym katalogu wyszuka w tym katalogu i jego podkatalogach wszystkie pliki z rozszerzeniem `*.mp3`, zaprezentuje użytkownikowi listę utworów w formacie

numer) *tytuł-albumu* (*wykonawca*): *tytuł-utworu*

i będzie oczekiwał na wybór jednego z nich, a następnie odtworzy wybrany utwór (za pomocą polecenia `mplayer(1)`, `play(1)`, `alsaplayer(1)`, `mpg123(1)` lub podobnego). Wybór i odtwarzanie utworów powinno być powtarzane, aż użytkownik wprowadzi znak EOF (`<ctrl>-D`), np.

```
$ mp3play
1) Nosferatu (Popol Vuh): Brothers in Darkness
2) Stefano Landi (Marco Beasley): Homo Fugit Velut Umbra
3) Piazzolla (Aquiles Delle-Vigne): Adios Nonino
4) Swan Lake (Pyotr Ilyich Tchaikovsky): Act I. Finale
5) Angst (Klaus Schulze): Freeze
Choose a number to play> 4
                              ... odtwarzanie utworu nr 4...
Choose a number to play> x
Wrong answer
Choose a number to play> 2
                              ... odtwarzanie utworu nr 2...
Choose a number to play> ^D
$
```

Uwagi i przypadki brzegowe:

- Wybór opcji można wygodnie zrealizować za pomocą instrukcji `select`.
- Nazwy plików mogą zawierać spacje.
- Skrypt powinien się właściwie zachować w przypadku wprowadzenia niewłaściwej odpowiedzi.
- Nazwa pliku, to nie wszystko. Zastanów się, jak ważne są metadane zapisane w plikach różnych formatów (mp3, epub, pdf). Czy we własnym życiu poświęcasz im dostatecznie wiele uwagi?
- Inne programy przetwarzające tagi ID3, to `id3`, `id3tool`, `id3v2`, `kid3` (okienkowy, obsługuje ID3 2.0, ale zależy od KDE) `kid3-qt` i `kid3-cli` (nie zależą od KDE, ale od Qt), `exiftool`, `btag`, `easytag`, `exfalso`, `extract`, `id3ren` oraz `mp3rename`.

Możliwe rozszerzenia:

- Prezentowanie listy posortowanej według albumów, autorów bądź tytułów.

Zadanie 2 (1 pkt). Statyczna konfiguracja systemów plików jest zapisana przez administratora w pliku `/etc/fstab`, zob. `fstab(5)`. Zaprogramuj skrypt `getdev`, który uruchomiony z argumentem będącym ścieżką dostępu do punktu montowania wypisze informacje o urządzeniu blokowym skonfigurowanym do montowania w danym miejscu wraz z parametrami montowania, np. jeśli w pliku `/etc/fstab` znajduje się wiersz

```
UUID=b121b129-3402-5fd3-ea55-431f03747cc9 /boot ext2 defaults 0 2
```

to polecenie `getdev /boot` wypisze

```
Device:           UUID=b121b129-3402-5fd3-ea55-431f03747cc9
Filesystem type:  ext2
Mount options:    defaults
Dump freqency:    0
Fsck pass number: 2
```

Uwagi i przypadki brzegowe:

- Skrypt powinien właściwe reagować na komentarze i puste wiersze w pliku `/etc/fstab`.
- Znak ukośnika (jeden lub nawet więcej) na końcu nazwy katalogu jest opcjonalny. Skrypt powinien wyszukiwać właściwie urządzenia niezależnie od tego, czy ukośnik występuje na końcu nazwy katalogu w pliku `/etc/fstab` lub w parametrze wywołania, czy też nie.
- Skrypt powinien się sensownie zachować w przypadku wywołania bez parametru.

Zadanie 3 (1 pkt). Sito Eratostenesa można zgrabnie zaprogramować współbieżnie wykorzystując model *obliczeń sterowanych przepływem danych*. Najpierw uruchamiamy proces, który wypisuje do swojego standardowego wyjścia kolejne liczby naturalne od `2` do `MAX`, po jednej w wierszu. Ten proces jest połączony potokiem z „filtrem”, który działa następująco: odczytuje ze standardowego wejścia jedną liczbę, zapamiętuje ją i wypisuje do standardowego wyjścia. Następnie uruchamia swoją kopię i łączy się z nią potokiem, po czym czyta ze swojego wejścia kolejne liczby, wybiera te, które nie dzielą się przez „jego” liczbę i wysyła je poprzez potok do swojego „młodszego wcielenia” (które postępuje oczywiście dokładnie tak samo).

Zaprogramuj skrypt `primes.sh`, który wypisuje do standardowego wyjścia wszystkie liczby pierwsze nie większe niż liczba przekazana mu jako parametr wywołania (a w razie jego braku nie większe niż 1000).

```
$ bash primes.sh 200 | column -x
2       3       5       7       11      13      17      19      23      29
31      37      41      43      47      53      59      61      67      71
73      79      83      89      97      101     103     107     109     113
127     131     137     139     149     151     157     163     167     173
179     181     191     193     197     199
```

Uwagi i przypadki brzegowe:

- Warto skorzystać z *funkcji*. Funkcje mogą być w bashu rekurencyjne i mogą być uruchamiane jako podprocesy.
- Proces, który utworzył potok, czeka na zakończenie wszystkich podprocesów uruchomionych w potoku, dlatego najprostsza implementacja jest bardzo nieefektywna: aby wypisać 1000 liczb pierwszych, musi *jednocześnie* działać 1000 procesów! Rozmiar bufora w potoku `p|q` wynosi zwykle 4 KiB (zob. polecenie wbudowane basha `ulimit -p`), jeśli więc proces `p` ma do wypisania nie więcej niż 4 KiB, to warto uruchomić proces `q` *asynchronicznie*, tj. tak: `p|q&`. Wówczas proces, który uruchomił potok zakończy działanie, gdy tylko skończy się proces `p`. Trzeba jeszcze zadbać, by cały skrypt nie zakończył działania *zanim* wszystkie liczby nie zostaną wypisane.
- Wśród standardowych narzędzi linuksowych nie ma odpowiednika powyższego programu. Warto jednak zapoznać się z programami `expr(1)` i `factor(1)` z pakietu GNU Coreutils.
- Wyniki działania różnych programów wygodnie można przeformatowywać za pomocą programów `column(1)` z pakietu BSD Mainutils oraz `fmt(1)` i `fold(1)` z pakietu GNU Coreutils.

Zadanie 4 (1 pkt). Rozwiąż poprzednie zadanie w nieco bardziej klasyczny sposób — napisz skrypt, który utworzy tablicę indeksowaną kolejnych liczb pierwszych, zgodnie z algorytmem:

PRIMES[0] = 2
$i \leftarrow 1$
$n \leftarrow 3$
while $n \leq$ MAX **do**
  $j \leftarrow 0$
  while PRIMES[j] $\times$ PRIMES[j] $\leq n$ **do**
    if n mod PRIMES[j] = 0
      then goto *next*
    fi
    j++
  done
  PRIMES[i++] $\leftarrow n$
next:
  n++
done
print PRIMES[0..$i-1$]

Zauważ, że w bashu nie ma instrukcji skoku, ale instrukcje `break` i `continue` mogą mieć parametr.

Zadanie 5 (1 pkt). Przeczytaj stronę podręcznika systemowego `proc(5)` omawiającą pseudosystem plików `procfs`. Zapoznaj się w szczególności z opisem zawartości pliku `/proc/stat`. Napisz skrypt `watch-cpu`, który będzie co sekundę wypisywał wiersz postaci

```
CPU: 1.8% user, 0.7% system, 97.3% idle, 0.2% iowait
```

Na końcu wiersza zamiast znaku `\n` skrypt powinien wypisywać znak `\r`, dzięki temu użytkownik będzie mieć wrażenie animacji tekstu (po wypisaniu wiersza kursor wraca na jego początek, a sekundę później w tym samym miejscu jest wypisywany zaktualizowany wiersz). Naciśnięcie dowolnego klawisza powinno natychmiast zakończyć działanie skryptu. Zadbaj o to, żeby tekst był zawsze poprawny oraz żeby skrypt nie pozostawiał po sobie śmieci na ekranie.

Zadanie 6 (1 pkt). Program `motion` to demon, który analizuje obraz z kamery (domyślnie z urządzenia `/dev/video0`, zob. *Video for Linux*) i włącza nagrywanie do pliku (domyślnie w katalogu `/var/lib/motion/`) w razie wykrycia ruchu. Wraz z upływem czasu sumaryczny rozmiar plików `*.mp4` w katalogu `/var/lib/motion/` staje się bardzo duży. Warto więc ustalić limit ilości zajętego miejsca w tym katalogu i np. raz na dobę usuwać lub przenosić na zewnętrzny dysk kilka najstarszych plików

(według daty ostatniej modyfikacji) tak, by sumaryczny rozmiar pozostałych plików nie przekraczał ustalonego limitu. Napisz skrypt `filerotate` (por. `logrotate(1)`), który wywołany z dwoma parametrami: limitem rozmiaru w bajtach i ścieżką dostępu do katalogu wypisze do standardowego wyjścia listę najstarszych plików, po jednym w wierszu, których usunięcie jest konieczne, by sumaryczny rozmiar pozostałych plików nie przekraczał limitu. Uwagi i problemy do rozważenia:

- Czy w podanym katalogu mogą być podkatalogi lub linki symboliczne?
- Czy nazwy plików mogą zawierać spacje?

Zadanie 7 (1 pkt). Jeśli musimy skopiować duży plik lub katalog, to może się on nie zmieścić w wybranym systemie plików. Polecenie `du(1)` podaje łączny rozmiar zbioru plików. Polecenie `df(1)` ujawnia zamontowane systemy plików oraz, m. in., ilość wolnego miejsca w tych systemach. Napisz skrypt `where-fits`, którego argumentami powinny być nazwy plików i katalogów, który wypisze listę zamontowanych systemów plików, w których podane pliki i katalogi się (jednocześnie) zmieszczą. Uwagi i przypadki brzegowe:

- Skrypt powinien właściwie obsługiwać spacje w nazwach plików.
- Ze względu na rezerwację przestrzeni dla użytkownika root oraz *quota* odpowiedź może nie być dokładna.

Zadanie 8 (1 pkt + bonus). Zaprogramuj w C maksymalnie uproszczoną powłokę systemową. Program powinien wczytywać wiersz ze standardowego wiersza i dzielić go na słowa oddzielone znakami spacji i tabulacji. Pierwsze ze słów powinno być pełną ścieżką dostępu do pliku wykonywalnego. Pozostałe słowa, to parametry wywołania tego programu. Powłoka powinna wywołać funkcję `fork` w celu utworzenia procesu potomnego, a uruchomiony proces potomny powinien przygotować odpowiednią tablicę `argv` i wywołać funkcję `execve`. Zob. np. Stephen Brennan, *Tutorial — Write a Shell in C*, 16 January 2015.[1]

Możesz następnie rozszerzyć swoją powłokę o (po 1 pkt. za każde rozszerzenie):

8a) wprowadzanie wiersza za pomocą biblioteki `readline`,

8b) wyszukiwanie ścieżki dostępu do polecenia według zmiennej `$PATH`,

8c) przekierowania,

8d) rozwijanie wzorców nazw plików,

8e) uruchamianie poleceń w tle i potoki.

[1] `https://brennan.io/2015/01/16/write-a-shell-in-c/`