Kurs administrowania systemem Linux 2025

Lista zadań na pracownię nr 4

Na zajęcia 19 i 20 marca 2025

Zadanie 1 (1 punkt). Przypomnijmy, że rozwinięcia są w Bashu wykonywane w następującej kolejności:

1. rozwinięcia nawiasów wąsatych, np. `file{1,2,3}`, `file{1..10}`,
2. rozwinięcia tyldy, np. `~/Downloads/`,
3. rozwinięcia zmiennych, np. `$HOME`,
4. podstawienia instrukcji, np. `$(cat file.txt)`,
5. podstawienia procesów, np. `<(pdftops file.pdf -)`,
6. rozwinięcia arytmetyczne, np. `$((N+1))`,
7. (powtórny) podział na słowa,
8. rozwinięcia nazw plików (*globów*), np. `file?-*.txt`.

Wszystkie z wyjątkiem powtórnego podziału na słowa (bo pierwszego podziału dokonuje lekser przed parsowaniem) są implementacjami różnych konstrukcji językowych. Podaj przykłady, czemu powtórny podział na słowa jest potrzebny. Podaj przykłady, czemu powinien być wykonany po a) rozwinięciach zmiennych, b) podstawieniach instrukcji. Podaj przykłady, czemu rozwinięcia instrukcji są wykonywane po rozwinięciach zmiennych. Dla wszystkich powyższych przypadków podaj też kontrprzykłady, że odwrotna kolejność byłaby niewygodna lub bezsensowna.

Zadanie 2 (1–4 punktów). Rozważany w zadaniu 1 z poprzedniej listy program `mp3play` wypisywał listę utworów w konsoli tekstowej (wykorzystując np. instrukcję `select` powłoki). W przypadku dużej liczby utworów takie rozwiązanie jest niewygodne. Wolelibyśmy, by lista utworów została wyświetlona w oknie (na pełnoekranowej konsoli tekstowej lub w środowisku graficznym) tak, by użytkownik mógł nawigować wśród utworów za pomocą klawiszy strzałek oraz klawiszy `<PgUp>`, `<PgDn>`, `<Home>`, `<End>` itp., wyszukiwać tytuły na te liście wpisując kilka początkowych liter itp. oraz zatwierdzać wybór klikając na odpowiedni przycisk (w środowisku graficznym), bądź naciskając klawisz `<Enter>`.

Wiele programów pozwala na bardzo łatwe tworzenie okien dialogowych zarówno w konsoli tekstowej (np. `whiptail`, `dialog`) jak i w środowisku graficznym (np. `zenity`). Po jednym punkcie za każde rozwiązanie:

a) Użyj `whiptail`.

b) Użyj `dialog`.

c) Użyj `zenity`.

d) Wypróbuj też `gxmessage`, `kdialog`, `yad`, `smenu` lub inny podobny program, według uznania i własnych preferencji.

Zadanie 3 (1 pkt). Niezbyt rozgarniętego początkującego użytkownika Linuksa irytuje konieczność poprzedzania nazwy programu ciągiem `./`, jeśli chce uruchomić program znajdujący się w bieżącym katalogu. Dodał zatem do swojego pliku `~/.profile` wiersz postaci

```
PATH=.:$PATH
```

Przygotuj archwium `lolcats.tar` zawierające kilka zdjęć kotów wraz z niewielkimi dodatkami, tak by wykonanie ciągu poleceń

```
$ tar xf lolcats.tar
$ ls
```

wyrządziło użytkownikowi *wielką krzywdę*. Dobrym kandydatem na *wielką krzywdę* jest wypisanie tekstu „You've been pwnd!" (polskie tłum.: „mam cię") lub uruchomienie programu `sl` (w Debianie instalowanego przez pakiet o tej samej nazwie).

Zadanie 4 (1 pkt). Opracuj (możesz się inspirować istniejącymi rozwiązaniami) inne eksploity podobne do opisanego w zadaniu 3 (np. *privilege escalation* wykorzystujący względne ścieżki do programów wywoływanych w skryptach, programy wykorzystujące SUID, nadużycia `sudo` itp.). *Uwaga:* przesłanie użytkownikowi informacji, że uruchomienie polecenia `rm -rf ~` powoduje wyświetlenie atrakcyjnych *lolcatów* oraz *Albański Wirus Komputerowy* mimo wielkiej siły rażenia nie są racjonalnymi eksploitami, gdyż odwołują się wyłącznie do socjotechniki i nie wykorzystują żadnych podatności technicznych.

Zadanie 5 (1 pkt). Wykorzystaj program `getopt(1)` w skrypcie `hwb`, którego zadaniem jest wypisywanie na ekranie pozdrowień typu „Hello, world!". Dla każdego argumentu *`imię`* program powinien wypisać wiersz postaci

`Hello, `*`imię`*`!`

Program powinien obsługiwać przynajmniej następujące opcje:

`-c`, `--capitalize` wypisanie imienia bądź słowa `world` wielką literą;

`--color=`[`never` | `auto` | `always`] kolorowanie imion (nigdy, tylko gdy standardowy strumień wyjściowy jest konsolą, zawsze), por. podobną opcję programu `ls(1)`;

`-g` *`text`*, `--greeting=`*`text`* zastąpienie słowa `Hello` podanym tekstem;

`-h`, `--help` wypisanie krótkiej ściągi;

`-v`, `--version` wypisanie nazwy i wersji programu oraz copyrightu;

`-w`, `--world` wypisanie dodatkowo wiersza `Hello, world!`

Zadanie 6 (1 pkt). Wykorzystując funkcję `getopt_long(3)` zaprogramuj w C program `hwc` działający podobnie do skryptu z zadania 5.

Zadanie 7 (1 pkt). Napisz stronę podręcznika `hwb(1)`. Zastosuj konwencje opisane w rozdziale DESCRIPTION na stronie `man(1)` (w szczególności umieść co najmniej rozdziały NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, EXAMPLE, AUTHORS i SEE ALSO oraz przestrzegaj konwencji zapisywania niektórych fraz kursywą i pismem pogrubionym).

Zadanie 8 (1 pkt). Przygotuj dokumentację `hwb.texi` zgodnie z konwencjami dokumentacji GNU. Wygeneruj wersje `info`, `html` i `pdf`.

Zadanie 9 (1 + 5 pkt. bonusowych). Napisy zawierające znaki narodowe zamienia się na ciągi znaków ASCII stosując jedną z dwóch technik: 1) pomijanie znaków spoza kodowania ASCII, 2) usuwanie znaków diakrytycznych (´ , ` , ¨ itp.) i zastępowanie znaków nieposiadających łacińskich odpowiedników specjalnymi znakami, np. `?`. Pierwszy sposób prowadzi często do zupełnie nieczytelnych napisów (np. `Za gl ja wia` — „Zażółć gęślą jaźń żółwia", `d` — „Łódź" itp.) lub wyjątkowo niezręcznych sytuacji (np. serwis `ikea.pl` generuje nazwy plików z instrukcją montażu mebli usuwając polskie znaki z nazw mebli i dodając rozszerzenie `.pdf`, co nie jest zbyt fortunne w przypadku takich artykułów, jak „płyta główna"). Drugi sposób jest nieco lepszy, czasem jednak też prowadzi do niejednoznaczności (np. „połowa kółka, pięć gosposi i kąt", „okaż mi łaskę" itp.).

Napisz skrypt `ascify`, który uruchomiony w danym katalogu wyszuka nazwy zawierające znaki spoza zbioru ASCII oraz znaki mające specjalne znaczenie dla powłoki (w tym metaznaki, znaki używane we wzorcach nazw plików itp.), zaproponuje dla każdego z nich zmianę na nazwę nie zawierającą wyżej wymienionych znaków, np.

`Np. zażółcić gęślą jaźń żółwia?.pdf` → `Np_zazolcic_gesla_jazn_zolwia.pdf`

i w razie akceptacji użytkownika wykona tę zmianę. Uwagi i przypadki brzegowe:

- Skrypt powinien radzić sobie z kolizjami nazw, np. dodając na końcu nazwy unikatowy numer.
- Warto rozważyć opuszczanie znaków interpunkcyjnych, kompresowanie ciągów spacji itp.
- Rozważ wykorzystanie programów `iconv(1)` i `sed(1)`.

Rozszerzenia (1 punkt bonusowy za każde):

a) Opcje:

`-y`, `--yes` zmiana nazw będzie wykonana bez pytania użytkownika o zgodę;
`-n`, `--dry-run` wypisuje proponowane nazwy, ale nie pyta użytkownika o akceptację i nie dokonuje zmiany;
`-h`, `--help` wypisanie krótkiej ściągi;
`-v`, `--version` wypisanie nazwy i wersji programu oraz copyrightu;
`-d`, `--directory` wykonanie zmiany nazw także dla nazw katalogów;
`-r`, `--recursive` wykonanie zmiany nazw także dla plików znajdujących się w podkatalogach;
`-e` *`enc`*, `--encoding=`*`enc`* interpretacja oryginalnej nazwy według podanego kodowania *`enc`*.

Domyślnym kodowaniem znaków powinno być UTF-8.

b) Umożliwienie ręcznej edycji zaproponowanej nazwy: „yes/no/edit”.

c) Sensowna transliteracja cyrylicy, np.

Батогов Е. В.: Gentoo Linux. Сборник статей.pdf →
`Batogov_Je_V_Gentoo_Linux_Sbornik_statjej.pdf`

d) Strona podręcznika `ascify(1)` dla programu.

e) Dokumentacja `texi` programu. Wygeneruj wersje `info`, `html` i `pdf`.

Uwaga: Pliki skopiowane z Internetu (por. powyższy przykład dokumentacji Gentoo) są zapisywane przez przeglądarkę pod oryginalnymi nazwami. Program `ascify(1)` bardzo by się przydał do uporządkowania skopiowanych plików. Postaraj się napisać go tak, żeby faktycznie był użytecznym narzędziem. Prowadzący bardzo go potrzebuje.