1 Teoria

1.1 Znak zachęty

Znak zachęty to komunikat mówiący o gotowości do przyjęcia polecenia od użytkownika, zazwyczaj jest to znak dolara `$`.

Na serwerze `wierzba` jest bardziej rozbudowany (`kwrobel@wierzba:~/public_html/$`):

- nazwa użytkownika
- nazwa hosta
- aktualna ścieżka

1.2 Automatyczne uzupełnianie - `Tab`

Klawisz `Tab` odpowiada za automatyczne uzupełnianie poleceń i ścieżek. Po wpisaniu początkowych znaków polecenia naciskamy `Tab`. Jeśli istnieje tylko jedno możliwe zakończenie polecenia to jest ono uzupełniane. W przeciwnym wypadku naciskamy klawisz `Tab` ponownie i zostaje wypisana lista możliwych zakończeń polecenia.

1.3 Historia

Za pomocą klawiszów strzałka w górę i w dół możemy poruszać się po wydanych poleceniach.

1.4 Ścieżki

Ścieżki do plików lub katalogów mogą być:

- bezwzględne - zaczynają się od korzenia `/`, np. `/home/epi/00_student`
- względne (względem aktualnego katalogu), np. `podkatalog/plik`
  - `.` - oznacza aktualny katalog
  - `..` - oznacza katalog wyżej, np. `../../pracow/kwrobel`

Znak tylda ~ jest automatycznie rozwijany do ścieżki katalogu domowego, np. `~/public_html` jest równoznaczny z `/home/pracow/kwrobel/public_html` (gdy katalogiem domowym użytkownika jest `/home/pracow/kwrobel`).

1.5 Wildcardy

`*`	Zastępuje dowolny ciąg znaków
`?`	Zastępuje jeden znak

1.6 `man`

Wyświetla podręcznik za pomocą programu `less`.

Argument	**Opis**
polecenie	dotyczy podanego polecenia

1.7 `ls`

Wypisuje zawartość katalogu.

Argument	Opis
ścieżka	odnosi się do podanej ścieżki, domyślnie aktualny katalog
`-a`, `--all`	wypisuje również pliki ukryte (zaczynające się od kropki)
`-l`	wypisuje dodatkowe informacje o plikach, np. prawa dostępu, właściciela, rozmiar, datę modyfikacji
`-R`, `--recursive`	wypisuje podkatalogi rekurencyjnie
`-h`, `--human-readable`	w połączeniu z argumentem `-l` rozmiar podawany jest w "ludzkich" jednostkach

1.8 `cd`

Zmienia katalog.

Argument	Opis
brak	przechodzi do katalogu domowego użytkownika
ścieżka	przechodzi do podanego katalogu
`-`	przechodzi do poprzedniego katalogu

1.9 `mkdir`

Tworzy katalog.

Argument	Opis
ścieżka	tworzy katalog o podanej nazwie
`-p`	tworzy katalogi rodziców

1.10 `echo`

Wypisuje podany napis (na standardowe wyjście).

Argument	Opis
napis	
`-n`	nie dodaje na końcu znaku nowej linii
`-e`	interpretuje znaki specjalne
`-E`	nie interpretuje znaków specjalnych (domyślne)

1.10.1 Znaki specjalne

Ciąg	Opis
`\\`	odwrócony ukośnik \
`\n`	nowa linia
`\r`	powrót karetki
`\t`	tabulacja

1.11 `cat`

Wypisuje zawartość pliku (na standardowe wyjście).

Argument	Opis
brak	wypisuje dane podane na standardowym wejściu, `Ctrl+D` (EOF - end of file) kończy wprowadzanie danych z klawiatury
ścieżki	wypisuje pliki o kolejnych ścieżkach

1.11.1 Przekierowania

`polecenie > ścieżka`	Zapisuje standardowe wyjście polecenia do pliku pod wskazaną ścieżką
`polecenie >> ścieżka`	Dopisuje standardowe wyjście polecenia do pliku pod wskazaną ścieżką
`polecenie1 | polecenie2`	Przekazuje standardowe wyjście polecenia 1 na standardowe wejście polecenia 2
`polecenie < ścieżka`	Przekazuje zawartość pliku pod wskazaną ścieżką na standardowe wejście polecenia

1.12 `rm`

Usuwa plik/katalog.

Argument	Opis
ścieżka	usuwa podany plik/katalog
`-r`	argument konieczny do usunięcia katalogu z zawartością
`-i`	generuje pytanie o usunięcie każdego pliku
`-f`	nie pyta o potwierdzenie usunięcia, dotyczy plików ukrytych

1.13 `cp`

Kopiuje plik/katalog.

Argument	Opis
ścieżka_skąd ścieżka_dokąd	kopiuje plik z jednej lokalizacji do drugiej
`-r`	argument konieczny do skopiowania katalogu

1.14 `mv`

Przenosi plik/katalog.

Argument	Opis
ścieżka_skąd ścieżka_dokąd	przenosi plik z jednej lokalizacji do drugiej

1.15 more

Wyświetla zawartość pliku używając paginacji. Zaleca się używanie programu `less`: "Users should realize that `less(1)` provides `more(1)` emulation plus extensive enhancements."

Argument	Opis
ścieżka	wyświetla zawartość podanego pliku

1.16 less

Umożliwia przeglądanie zawartości plików w konsoli.

Argument	Opis
ścieżka	wyświetla zawartość podanego pliku

Wyszukiwanie odbywa się przez wciśnięcie ukośnika `/` i podanie frazy. Klawisz `N` odpowiada za wyszukanie następnego wystąpienia a kombinacja `Shift+N` za poprzedniego. Z programu wychodzimy naciskając `Q`.

1.17 find

Wyszukuje pliki.

Argument	Opis
ścieżka `-name` wzorzec	wyszukuje pliki w podanym katalogu o podanym wzorcu

1.18 ln

Tworzy dowiązanie (skrót) do pliku/katalogu.

Argument	Opis
ścieżka_celu nazwa	tworzy dowiązanie do celu o podanej nazwie
`-s`	tworzy dowiązanie symboliczne

1.19 pwd

Wypisuje bezwzględną ścieżkę aktualnego katalogu.

1.20 dirname

Wypisuje ścieżkę do katalogu na podstawie podanej ścieżki (nie operuje na dysku).

Argument	Opis
ścieżka	analizuje podany ciąg znaków

1.21 basename

Wypisuje nazwę pliku na podstawie podanej ścieżki (nie operuje na dysku).

Argument	Opis
ścieżka	analizuje podany ciąg znaków
ścieżka rozszerzenie	analizuje podany ciąg znaków oraz usuwa rozszerzenie

1.22 `touch`

Tworzy pusty plik.

Argument	Opis
ścieżka	tworzy pusty plik pod podaną ścieżką

2 Praktyka

Warning
Proszę czytać wszystkie komunikaty wyświetlane w konsoli.

2.1 Zadanie 0

Important
`Tab`

W ilu najmniej naciśnięciach klawiszy da się wydać polecenie `mkdir /etc/passwd` (tak, nie ma ono sensu)?

2.2 Zadanie 1

Important
`ls, cd, mkdir`

W katalogu domowym (`/home/epi/login`) utwórz następującą strukturę katalogów:

- `~/dokumenty`
  - `praca/`
    - `raporty/`
  - `dom/`
    - `zakupy/`
  - `studia/`
    - `podania/`

2.3 Zadanie 2

Spróbuj wykonać powyższe zadanie używając tylko jednego polecenia zamiast sekwencji `mkdir, ls, cd`.

2.4 Zadanie 3

Important
`echo`

W katalogu `zakupy/` utwórz plik o nazwie `lista`. Zrób to za pomocą komendy: `echo mleko > lista`

2.5 Zadanie 4

Important
`cat`

Wyświetl zawartość pliku lista.

2.6 Zadanie 5

Zaobserwuj co się stanie jeśli wykonasz następnie polecenie `echo chleb > lista`

2.7 Zadanie 6

Important
`rm`

Usuń plik lista i przetestuj czym różni się poprzednia komenda od użycia komend:

```
echo mleko >> lista
echo chleb >> lista
```

2.8 Zadanie 7

Usuń katalog `dom/` wraz z podkatalogami i plikami. Spróbuj wykonać to także tylko jednym poleceniem.

2.9 Zadanie 8

Important
`cp, mv`

Do katalogu `raporty/` skopiuj plik `/etc/passwd` (który zawiera listę użytkowników w systemie). Plik w ścieżce docelowej powinien nazywać się `uzytkownicy`. Spróbuj skopiować plik jednocześnie nadając mu nową nazwę (jedna komenda).

2.10 Zadanie 9

Important
`more, less`

Zmniejsz okno terminala i spróbuj wyświetlić zawartość pliku `uzytkownicy`. Zawartość nie zmieściła się w jednym oknie. Spróbuj wykonać polecenia:

```
cat uzytkownicy | more #(bardziej poprawne to more uzytkownicy)
cat uzytkownicy | less #(bardziej poprawne to less uzytkownicy)
```

2.11 Zadanie 10

Important
`find`

Spróbuj wyszukać za pomocą polecenia `find` wszystkie pliki w Twoim katalogu domowym zaczynające się na literę `d`. W tym celu użyj wildcard `*`

2.12 Zadanie 11

Important
`ln`

W katalogu `studia/` utwórz katalog `programy/` w którym utwórz katalogi `program-1.0`, `program-1.1`, `program-1.3`. Zrób link symboliczny o nazwie `program` do katalogu z najwyższą wersją (1.3). Spróbuj dodać plik do katalogu `program-1.3` i przekonaj się, że plik ten znajduje się także w katalogu `program`.

2.13 Zadanie 12

Sprawdź poleceniem `ls -al`, że plik `program` jest rzeczywiście linkiem symbolicznym i wskazuje na katalog `program-1.3`.

2.14 Zadanie 13

Important
`pwd, dirname, basename`

Na przykładzie pliku `uzytkownicy` zaprezentuj działanie programów `pwd`, `dirname`, `basename`. Odczytaj pełną ścieżkę za pomocą `pwd`, a następnie podaj ją jako argument do dwóch pozostałych programów.

2.15 Zadanie 14

Na końcu usuń cały katalog `dokumenty/`.

3 Tematy dodatkowe

1. Zainteresuj się czym różni się link twardy (hard link) od symbolicznego.
2. Sprawdź do czego służą polecenia: `bc`, `cal`, `df`, `diff`, `du`, `finger`, `grep`, `host`, `htop`, `id`, `ifconfig`, `kill`, `lynx`, `md5sum`, `nano`, `nslookup`, `pico`, `pine`, `ping`, `rmdir`, `reset`, `scp`, `skill`, `su`, `sudo`, `top`, `uname`, `unzip`, `unrar`, `vi`, `vim`, `wc`, `wget`, `who`, `whoami`, `whois`, `write`, `yes`, `zip`

1 Teoria

System Linux wywodzi się z rodziny systemów unixowych, których przeznaczeniem było głównie działanie jako systemy serwerowe obsługujące bardzo wielu użytkowników. Z tego powodu w systemie Linux istnieje stosunkowo rozbudowany system zarządzania użytkownikami. Każdy użytkownik rozpoznawany jest po nazwie użytkownika oraz jednocześnie w systemie po specjalnym numerze `UID`, który jest unikalny dla niego. Dodatkowo użytkownicy mogą przypisani być do grup. Grupy także rozpoznawane są za pomocą nazwy oraz specjalnego unikalnego numeru `GID`. Grupa powinny reprezentować pewną wspólna cechę wielu użytkowników. Głównym użytkownikiem systemu jest `root` który ma zawsze `UID` równy `0`. Jest on także członkiem grupy `root`. Nazwy grup i użytkowników mogą się powtarzać, co jest w niektórych dystrybucjach domyślną polityką tworzenia grup dla użytkowników (nazwa grupy taka sama jak login).

1.1 Prawa dostępu do plików

1.1.1 Kategorie

Prawa dostępu do pliku definiujemy dla 3 kategorii:

- **u**ser - użytkownik, czyli właściciel pliku
- **g**roup - grupa przypisana do pliku
- **o**ther - wszyscy pozostali

1.1.2 Rodzaje praw dostępu

Wyróżniamy 3 rodzaje praw:

- **r**ead - odczyt zawartości pliku/katalogu ($4 = 100_2$)
- **w**rite - zapis do pliku/katalogu ($2 = 010_2$)
- e**x**ecute - wykonanie pliku/otwarcie katalogu ($1 = 001_2$)

Polecenie `ls -l` prezentuje prawa w zapisie symbolicznym: `-rwxr-xr--x fizyk fizyk 0 kwi 16 13:09 plik`

Pierwsza kolumna ma format `tuuugggooo`:

- `t` - typ pliku
  - `-` - zwykły plik
  - `d` - katalog
  - `l` - dowiązanie symboliczne
- `u` - uprawnienia użytkownika
- `g` - uprawnienia grupy
- `o` - uprawnienia wszystkich pozostałych

1.1.3 Sposoby reprezentacji praw

Typ zapisu	Odczyt	Zapis	Wykonanie
binarny	2^2	2^1	2^0
liczbowy	4	2	1
symboliczny	r	w	x

1.2 `chmod`

Zmienia prawa dostępu do pliku.

Argument	Opis
prawa ścieżka	zmienia prawa dostępu do podanego pliku
`-R`	rekurencyjnie ustawia prawa wszystkim plikom w podkatalogach

Prawa można podać w zapisie numerycznym lub symbolicznym.

1.2.1 Zapis numeryczny

Przy zapisie numerycznym dodajemy do siebie reprezentację liczbową dla każdej kategorii (osobno dla użytkownika, grupy i pozostałych), co daje nam 3 liczby, np `755`.

1.2.2 Zapis symboliczny

Zapis symboliczny pozwala również na zmianę dotychczasowych praw za pomocą znaków:

- = - ustawia podane prawa
- + - dodaje podane prawa
- − - usuwa podane prawa

Do każdej kategorii odwołujemy się przez odpowiednią literę (`u`,`g` lub `o`) stawiając za nim jeden z powyższych znaków i podając prawa, np. `u=rwx,g+rw,o-r`.

1.3 `sudo`

Uruchamia polecenie jako administrator.

Argument	Opis
polecenie	uruchamia podane polecenia jako administrator

1.4 `whoami`

Wypisuje nazwę aktualnego użytkownika.

1.5 `id`

Wypisuje informacje o użytkowniku i grupach, do których należy.

1.6 `adduser, addgroup`

Dodaje użytkownika/grupę do systemu. Na podstawie informacji w `/etc/adduser.conf` konfiguracja obejmuje m.in.:

- ustawienie powłoki na Bash
- utworzenie katalogu domowego użytkownika

Argument	Opis
nazwa	dodaje użytkownika/grupę o podanej nazwie
użytkownik grupa	dodaje użytkownika do grupy

1.7 /etc/passwd

`passwd` jest plikiem tekstowym z jednym rekordem na linię, z których każda opisuje jedno konto użytkownika. Każdy rekord (linia) składa się z siedmiu pól oddzielonych dwukropkami. Kolejność rekordów w pliku jest zazwyczaj nieistotna. Przykład: `jsmith:x:1001:1000:Joe Smith, pokój 1007, (234) 555-8910, (234) 555-0044, e-mail:/home/jsmith:/bin/sh`

Kolejne pola w rekordzie oznaczają:

1. Nazwa użytkownika
2. Drugie pole przechowuje informację używaną do sprawdzania hasła użytkownika. W nowych systemach wartość tego pola to "x", gdyż przechowywanie haseł, do których mają dostęp wszyscy użytkownicy nie jest bezpieczne. Obecnie stosuje się plik `/etc/shadow`. Ustawienie tego pola na gwiazdkę "*" wyłącza konto, aby zapobiec jego użyciu.
3. Identyfikator użytkownika `UID`, numer, który system operacyjny używa do celów wewnętrznych.
4. Identyfikator grupy `GID`. Liczba ta określa podstawową grupę użytkownika, wszystkie pliki, które są tworzone przez użytkownika są początkowo dostępne dla tej grupy.
5. Piąte pole, zwane polem GECOS, jest komentarzem, który opisuje osoby lub konta. Zazwyczaj jest to zbiór wartości oddzielonych przecinkami w tym pełnej nazwy użytkownika i dane kontaktowe.
6. Ścieżka do katalogu domowego użytkownika.
7. Domyślna powłoka (shell). Program, który jest uruchamiany przy każdym zalogowaniu do systemu. Dla użytkownika interaktywnego, zazwyczaj jest to jeden z systemu tłumaczy linii komend np. `bash`.

1.8 /etc/group

`group` jest to plik, w którym przechowywane są informacje o grupach. Tak jak w przypadku pliku `passwd` jeden rekord stanowi jedna linia rozdzielana znakiem dwukropka. Przykład: `cdrom:x:24:joe,admins,kate`

Poszczególne pola w rekordzie oznaczają:

1. Nazwa grupy
2. Pole hasła, przeważnie nie używane. Umożliwia tworzenie specjalnych uprzywilejowanych grup.
3. Identyfikator grupy `GID`.
4. Lista użytkowników grupy rozdzielona przecinkami. Wszyscy użytkownicy wymienieni w tym polu należą do danej grupy i zyskują jej uprawnienia.

1.9 su

Zmienia użytkownika, na którego jesteśmy zalogowani.

Argument	Opis
login	loguje się na podanego użytkownika

1.10 `chgrp`

Zmienia grupę, do której przypisany jest plik. Pozwala na zmianę grupy, na taką, do której należy użytkownik.

Argument	Opis
grupa ścieżka	przypisuje grupę do podanego pliku/katalogu

1.11 `chown`

Zmienia użytkownika i grupę, do której przypisany jest plik. Zazwyczaj tylko administrator ma prawa do użycia tej komendy.

Argument	Opis
[użytkownik][:grupa] ścieżka	przypisuje grupę/użytkownika do podanego pliku/katalogu

1.12 `deluser, delgroup`

Usuwa użytkownika/grupę.

Argument	Opis
nazwa	usuwa użytkownika/grupę o podanej nazwie
--remove-home użytkownik	usuwa użytkownika wraz z jego katalogiem domowym

1.13 `passwd`

Zmienia hasło użytkownika.

Argument	Opis
brak	zmienia hasło aktualnego użytkownika
login	zmienia hasło użytkownika o podanym loginie

2 Praktyka

2.1 Zadanie

Zapoznaj się z ideą wirtualizacji oraz narzędziem VirtualBox dostępnym na komputerach w laboratorium.

2.2 Zadanie

Pobierz i uruchom w VirtualBox obraz systemu Ubuntu z adresu: http://wierzba.wzks.uj.edu.pl/~kwrobel/SOS/Ubuntu9.ova

2.3 Zadanie

Important
`sudo`, `whoami`

Uzyskaj prawa użytkownika root wykonując np. polecenie `sudo bash`. Hasło użytkownika `ubuntu` to `reverse`, sprawdź poleceniem `whoami` czy posiadasz dostęp jako użytkownik `root`.

2.4 Zadanie

Important
`adduser`

Utwórz w systemie użytkowników: `marek`, `ania`, `jurek`.

2.5 Zadanie

Sprawdź jak zmienił się plik `/etc/passwd`.

2.6 Zadanie

Important
`addgroup`

Utwórz w systemie grupy: `marketing`, `zarzad`.

2.7 Zadanie

Sprawdź jak zmienił się plik `/etc/group`.

2.8 Zadanie

Dodaj do grupy `marketing` użytkowników `marek` i `ania`, a do grupy `zarzad` tylko użytkownika `ania`.

2.9 Zadanie

Important
`su`

Jako użytkownik `marek` utwórz katalog `~/projekt` a w nim plik tekstowy z dowolną treścią o nazwie `~/projekt/zalozenia`.

2.10 Zadanie

Important
`chmod`, `chgrp`

Ustaw takie prawa dostępu do tego pliku aby mogły go odczytać wszystkie osoby ale edytować tylko z grupy `marketing`. Sprawdź, logując się jako `marek` lub `ania`, czy możesz wyświetlić plik i go zmodyfikować, a jako `jurek` możesz tylko wyświetlić.

2.11 Zadanie

Ustaw takie prawa dla katalogu `projekt`, aby wyświetlić jego zawartość mogła tylko osoba z grupy `zarzad` (czyli tylko `ania`) - pozostałe prawa pozostają niezmienione. Sprawdź czy jako `marek` znając pełną ścieżkę pliku nadal możesz edytować plik `~/projekt/zalozenia`.

2.12 Zadanie

Ustaw takie prawa do katalogu projekt, aby wejść do katalogu mogła tylko osoba z grupy `zarzad`. Sprawdź czy jako `marek` nadal możesz edytować plik?

2.13 Zadanie

Dodaj kolejne pliki i katalogi do katalogu projekt (przynajmniej 2 poziomy). Spróbuj ustawić dowolne prawa dostępu rekurencyjnie wszystkim plikom i podkatalogom dla `~/projekt`. Np. prawa `666`.

2.14 Zadanie

Important
`deluser`, `delgroup`

Usuń grupy i użytkowników w systemie.

1 Teoria

Podczas zajęć omówione zostaną komendy związane z przeglądaniem stanu systemu.

Proces w systemie Linux jest to uruchomiona instancja programu binarnego. Taka uruchomiona instancja zajmuje pewne zasoby w pamięci, system operacyjny zarządza wskaźnikiem instrukcji i wykonaniem kolejnych kroków instrukcji procesu. Procesy posiadają specjalny unikatowy numer w systemie - PID (process ID). Numer ten przyznawany jest z reużytkowalnej puli, ale w danym momencie nie ma dwóch procesów o tym samym PID. Każdy proces uruchomiony jest z poziomu konkretnego użytkownika i dysponuje prawami takimi samymi jak ten użytkownik. Najważniejszym procesem w systemie jest `init` który ma zawsze PID `1`, a jego właścicielem jest użytkownik `root`.

1.1 Procesy a demony

Demony, inaczej usługi, są pewnym rodzajem programów, które są uruchamiane zazwyczaj przy starcie systemu działając najczęściej cały czas w tle. Procesy tych usług posiadają w Linux specjalne określenie: demon (ang. daemon). W praktyce wiele programów umożliwia uruchomienie ich w trybie demona, a typowymi usługami działającymi w ten sposób są: serwery WWW (np. Apache, nginx), DNS (np. bind), mail (pocztowe), X server, baza danych (np. MySQL, PostgreSQL), demon usług sieciowych i DHCP, itp... Procesy demonów będące usługami w każdej chwili mogą przyjąć od innego programu polecenie (żądanie) w celu ich obsłużenia. Demonem jest np. serwer HTTP, który cały czas jest uruchomiony w tle, a w momencie gdy przychodzi żądanie od przeglądarki wysyła odpowiedni plik lub wykonuje jakąś akcję. Demon może również obsługiwać programy działające na tym samym komputerze.

Kolejną cechą charakterystyczną demonów, jest to że nowoczesne dystrybucje Linux najczęściej posiadają wbudowane podsystemy do zarządzania uruchamianiem tych procesów. Np. dbają one o to aby przy każdym starcie systemu procesy demonów były automatycznie uruchomione. Często również do obsługi procesów demonów używa się różnego rodzaju aplikacji monitorujących stan ich procesów po to aby np. gdy demon zakończy się z błędem, był on znowu uruchomiony i dzięki temu było dalej możliwe łączenie się z daną usługą. Do zatrzymania działania demona służą specjalne skrypty, które w popularnych dystrybucjach (Debian, Ubuntu) znajdują się zazwyczaj w katalogu `/etc/init.d/`. W Ubuntu powinno się używać polecenia `service`, które je wywołuje.

Skrypty te mogą być uruchamiane z następującymi opcjami:

- start - uruchamia demona
- stop - zatrzymuje demona
- restart - zatrzymuje, i ponownie uruchamia demona; przydatne np. w przypadku gdy zmieniona została konfiguracja danego demona

Niektóre mogą przyjmować również inne opcje, jednak te trzy są standardowe.

1.2 Plik `/proc/cpuinfo`

Zawiera informacje na temat procesora.

1.3 Plik `/proc/meminfo`

Zawiera informacje na temat pamięci RAM w systemie.

1.4 `ps`

Wypisuje listę procesów w systemie.

Argument	Opis
brak	wypisuje procesy związane z terminalem

Argument	Opis
`-l`	dokładniejsze informacje
`-A`	wypisuje wszystkie procesy w systemie
`-u` użytkownik	wypisuje procesy danego użytkownika
`-f`	wypisuje również argumenty poleceń

1.5 `top`

Wyświetla procesy w trybie interaktywnym.

Sortowanie odbywa się domyślnie po kolumnie %CPU. Zmiana sortowania odbywa się przez naciśnięcie > i <.

Argument	Opis
`-d`	Określa opóźnienie między odświeżeniami ekranu. Można to zmieniać komendą interakcyjną `s`.
`-p`	Monitoruje jedynie procesy o danym PID.
`-S`	Określa tryb kumulacyjny, gdzie każdy proces jest wypisywany z czasem CPU, który spożytkowanym przez niego oraz jego martwe procesy potomne.
`-i`	Ignoruje wszelkie procesy zombie i procesy bezczynne.
`-c`	wyświetla linię poleceń zamiast samej nazwy polecenia.

1.6 Sygnały

(na podstawie/źródło: http://students.mimuw.edu.pl/SO/LabLinux/PROCESY/PODTEMAT_3/sygnaly.html)

Procesy komunikują się z jądrem systemu, a także między sobą aby koordynować swoją działalność. Linux wspiera kilka mechanizmów komunikacji zwanych IPC (Inter-Process Communication mechanisms). Jednym z nich są sygnały, zwane inaczej przerwaniami programowymi.

Sygnały mogą być generowane bezpośrednio przez użytkownika (funkcja `kill()`), może wysyłać je jądro oraz procesy między sobą (funkcja systemowa `kill()`) . Dodatkowo pewne znaki z terminala powodują wygenerowanie sygnałów. Na przykład na każdym terminalu istnieje tak zwany znak przerwania (ang. interrupt character) i znak zakończenia (ang. quit character). Znak przerwania (zazwyczaj `Ctrl+C` lub `Delete`) służy do zakończenia bieżącego procesu (wygenerowanie SIGINT). Wygenerowanie znaku zakończenia (zazwyczaj `Ctrl-\`) powoduje wysłanie sygnału SIGQUIT powodującego zakończenie wykonywania bieżącego procesu z zapisaniem obrazu pamięci.

Istnieją oczywiście pewne ograniczenia - proces może wysyłać je tylko do procesów mających tego samego właściciela oraz z tej samej grupy (te same UID i GID). Bez ograniczeń może to czynić jedynie jądro i administrator. Jedynym procesem, który nie odbiera sygnałów jest `init` (PID równy 1).

Sygnały są mechanizmem asynchronicznym - proces nie wie z góry kiedy sygnał może nadejść i głównym ich zadaniem jest informowanie procesu o zaistnieniu w systemie wyjątkowej sytuacji (np. spadek napięcia w sieci). Ponadto są wykorzystywane przez shelle do kontroli pracy swoich procesów potomnych.

Każdy z sygnałów posiada swoje znaczenie (określające w jakiej sytuacji powinien być wysłany) jak również związana jest z nim pewna domyślna akcja jaką system wykonuje przy jego obsłudze. Oto opis znaczenia poszczególnych sygnałów według numeracji w systemie Linux.

Nr	Nazwa	Opis
2	`SIGINT`	`Ctrl+C`, przerwij proces
3	`SIGQUIT`	zamknij proces i zapisz stan pamięci procesu
9	`SIGKILL`	zakończ proces natychmiast (nie może być przechwycony ani zignorowany)
15	`SIGTERM`	podobny do SIGINT
23	`SIGSTOP`	zatrzymuje proces do ponownego wznowienia (nie może być przechwycony ani zignorowany)
20	`SIGTSTP`	`Ctrl+Z`, w porównaniu do SIGSTOP może być obsłużony

1.7 `kill`

Polecenie służy do wysyłania sygnałów do procesów.

Użycie: `kill -SYGNAL PID`, gdzie `SYGNAL` to numer albo nazwa sygnału.

1.8 `killall`

Polecenie służy do wysyłania sygnałów do wielu procesów o tej samej nazwie.

2 Praktyka

2.1 Zadanie

Important
`/proc/cpuinfo`

Sprawdź jaki procesor znajduje się w komputerze w laboratorium oraz na serwerze wierzba. Zauważ, że dodatkowe rdzenie procesorów są reprezentowane jako kolejne procesory na liście.

2.2 Zadanie

Important
`/proc/meminfo`

Sprawdź ile pamięci RAM dysponuje komputer w laboratorium i serwer wierzba. Wartości podaj w MB oraz w GB dokonując odpowiednich przeliczeń.

2.3 Zadanie

Important
`ps`

Sprawdź jakie procesy zostały uruchomione w bieżącej sesji shella.

2.4 Zadanie

Sprawdź jakie procesy zostały uruchomione w całym systemie.

2.5 Zadanie

Sprawdź jakie procesy zostały uruchomione przez twojego użytkownika.

2.6 Zadanie

Important
`top`

Uruchom polecenie `top` i posortuj listę procesów po ilości czasu procesora (TIME), po obciążeniu procesora (%CPU), po zajętej pamięci (%MEM). Znajdź proces, który najbardziej obciąża procesor oraz proces który najbardziej obciąża pamięć. W tym celu możesz uruchomić dowolne programy na komputerze w laboratorium.

2.7 Zadanie

Utwórz prosty skrypt, którym będziesz testować działanie programów zarządzających procesami. W tym celu otwórz ulubiony edytor tekstu i przepisz poniższy kod:

```
#!/bin/bash
echo "Zasypiam na $1 sekund."
sleep $1
echo "Pobudka. Koniec."
```

Skrypt zapisz pod nazwą `spioch.sh`. Nadaj uprawnienia temu plikowi w taki sposób aby można było go uruchomić jako zalogowany użytkownik w systemie (dodaj uprawnienia executable).

2.8 Zadanie

Będąc w tym samym katalogu co plik `spioch.sh` spróbuj uruchomić skrypt w następujący sposób: `./spioch.sh 3`

Skrypt powinien zasnąć na 3 sekundy, a następnie powinien pojawić się napis `Pobudka.Koniec.`

2.9 Zadanie

Uruchom skrypt na wiele sekund (np. 300). Postaraj się zakończyć go przed czasem wciskając kombinację `Ctrl+C`. Jaki sygnał został wysłany do procesu? Czy napis `Pobudka.Koniec.` wyświetlił się na ekranie?

2.10 Zadanie

Uruchom ponownie skrypt na wiele sekund. Otwórz nową kartę terminala. Postaraj się odnaleźć go za pomocą komendy `ps` oraz `top`. Odnajdź PID tego procesu.

2.11 Zadanie

Important
`kill`

Znając numer PID procesu z uruchomionym skryptem śpiocha, spróbuj w osobnej karcie terminala wysłać do niego sygnały: SIGINT, SIGTERM, SIGQUIT, SIGKILL. Za każdym razem sprawdź co się stało z procesem śpiocha. Jeśli trzeba uruchom go ponownie.

2.12 Zadanie

Przy użyciu maszyny wirtualnej (VirtualBox) spróbuj uruchomić proces jako jeden użytkownik, a następnie zabić go jako inny użytkownik. Czy to możliwe? Próbę ponów jako użytkownik `root`.

1 Teoria

1.1 `wc`

Zlicza liczbę linii, słów i bajtów (znaków).

Argument	Opis
`-l`	Zlicza tylko liczbę linii.

1.2 `sort`

Sortuje linie pliku.

Argument	Opis
`-n`	Sortuje linie traktując je jako linie
`-r`	Sortuje w odwrotnej kolejności

1.3 `uniq`

Usuwa powtarzające się po sobie linie.

1.4 `cut`

Wycina fragmenty każdej linii.

Argument	Opis
`-f` numery pól	Wycina wybrane pola.
`-d` separator	Dzieli linię na pola używając podanego separatora.

1.5 `wget`

Pobiera zasób.

Argument	Opis
adres	Pobiera zasób znajdujący się pod danym adresem.

1.6 `head`

Wypisuje pierwsze 10 linii.

Argument	Opis
`-n` liczba	Wypisuje podaną liczbę pierwszych linii.

1.7 `tail`

Wypisuje ostatnie 10 linii.

Argument	Opis
`-n` liczba	Wypisuje podaną liczbę ostatnich linii.

1.8 `date`

Wypisuje aktualną datę.

Argument	Opis
+FORMAT	Wypisuje datę w określonym formacie (szczegóły w podręczniku).

1.9 `/dev/null`

Wirtualne urządzenie, które zapomina wszystko to co do niego przekierujemy

2 Praktyka

2.1 Zadanie

Important
`wc`

Spróbuj policzyć liczbę plików i katalogów w twoim katalogu domowym za pomocą polecenia `ls` i `wc`.

2.2 Zadanie

Policz ile użytkowników znajduje się w systemie Linux na komputerze w laboratorium.

2.3 Zadanie

Important
`cut`

Za pomocą komendy `cut` spróbuj obciąć ze strumienia listy użytkowników same loginy użytkowników.

2.4 Zadanie

Wykonaj to samo co w powyższym punkcie, ale wytnij ścieżkę standardowego shella użytkowników. (wskazówka, polecenie `cut` posiada dwie przydatne opcje `-f -d`)

2.5 Zadanie

Important
`sort`, `uniq`

Policz ile różnych shelli używają użytkownicy w systemie Linux na komputerze w laboratorium oraz na wierzbie?

2.6 Zadanie

Important
`wget`, `head`, `tail`

Pobierz plik z expose premiera (np. poleceniem wget):

```
wget http://wierzba.wzks.uj.edu.pl/~kwrobel/SOS/lab-05/expose_tuska.txt
```

a. Wyświetl zawartość pliku.
b. Użyj stronicowania `more` i `less`, aby przeglądnąć cały plik.
c. Za pomocą polecenia less odnajdź na ekranie wszystkie wystąpienia wyrazu `Polska`.
d. Wyświetl pierwsze 25 linii pliku.
e. Wyświetl ostatnie 13 linii pliku.
f. Wyświetl linie 10 do 15 pliku.
g. Policz liczbę wyrazów w pliku.
h. Dla każdej linii wyświetl 4 wyraz w tej linii.

2.7 Zadanie

Zapisz listę plików i katalogów znajdujących się w katalogu `/etc` do pliku `lista_etc`.

2.8 Zadanie

Użyj tego samego polecenia jak w punkcie powyżej ale zamiast `/etc` użyj nieistniejącego katalogu. Czy komunikat błędu zapisał się do pliku czy wyświetlił na ekranie?

2.9 Zadanie

Zmodyfikuj polecenie z powyższego punktu tak aby komunikat błędu także zapisał się do pliku.

2.10 Zadanie

Important
`date`

Napisz polecenie, które po każdym uruchomieniu dopisze obecną datę i godziną do pliku o nazwie `uruchomienia`.

1 Teoria

1.1 Tworzenie zmiennej

```
ZMIENNA="jakas wartosc"
```

1.2 Wartość zmiennej

W celu uzyskania wartości zmiennej poprzedzamy jej nazwę znakiem dolara, np.:

```
echo $ZMIENNA
```

1.3 Zmienne specjalne

Zmienna	Opis
`$?`	Zwraca kod zakończenia programu.
`$#`	Liczba argumentów, z którymi uruchomiono program.
`$N`	gdzie `N` to kolejna liczba naturalna - zawiera kolejne argumenty wywołania

1.4 Zmienne środowiskowe

Zmienna	Opis
`$USER`	Nazwa aktualnego użytkownika
`$HOME`	Ścieżka do katalogu domowego
`$PATH`	Zwraca listę ścieżek na których wyszukiwane są programy do uruchomienia, ścieżki oddzielone są za pomocą znaku dwukropka
`$RANDOM`	Zwraca losową liczbę z przedziału 0 i 32767.
`$PWD`	Ścieżka aktualnego katalogu
`$SHELL`	Ścieżka do powłoki

1.5 Zasięg zmiennych

Poleceniem `export` zmieniamy zasięg zmiennej na globalny.

1.6 Zmiana wartości zmiennej dla konkretnego programu

W celu podania zmiennej dla danego uruchomienia poprzedzamy polecenie definicją zmiennej.

```
ZMIENNA="wartosc" ./program.sh
```

1.7 Renderowanie zmiennej

Podobnie jak przy podawaniu formatu daty do polecenia `date`, możemy korzystać ze zmiennych w poleceniach.

```
echo "Ala ma $ZMIENNA"
ls $ZMIENNA.txt
```

1.8 Przypisanie `stdout` do zmiennej

```
ZMIENNA=`ls -al`
ZMIENNA=$(ls -al)
```

1.9 Wczytanie wartości zmiennej interaktywnie

```
read ZMIENNA
```

1.10 Wykonywanie operacji matematycznych

```
echo $((1+2))
echo $[1+2]
echo $[A+B]
```

1.11 Obcinanie części tekstu

`${ZMIENNA OPERATOR WZORZEC}` - bez odstępów wokół operatora

Operatory:

- `%` - obcina najkrótsze dopasowanie do wzorca od tyłu
- `%%` - obcina najdłuższe dopasowanie do wzorca od tyłu
- `#` - obcina najkrótsze dopasowanie do wzorca od przodu
- `##` - obcina najdłuższe dopasowanie do wzorca od przodu

2 Praktyka

2.1 Zadanie

Important
`set`, `export`

Wyświetl zawartość zmiennych systemowych.

2.2 Zadanie

Wyświetl numer użytkownika UID na komputerze w laboratorium i na wierzbie. Czy są takie same?

2.3 Zadanie

Ustaw zmienną środowiskową o nazwie `OPCJA` na wartość `-alh`

2.4 Zadanie

Użyj zmiennej środowiskowej `OPCJA` razem z poleceniem `ls` aby wyświetlić pliki w katalogu `$HOME` z opcjami znajdującymi się w zmiennej `$OPCJA`.

2.5 Zadanie

Important
`exit`

Sprawdź czy zmienna `OPCJA` dostępna jest w kolejnej instancji shella którą uruchomisz z bieżącej instancji (uruchom polecenie `bash`, aby wyjść z kolejnej instancji shella wpisz `exit`).

2.6 Zadanie

Sprawdź jaki kod wyjścia (exit status) zwraca polecenie `ls` gdy poprawnie wylistuje pliki oraz wtedy gdy podany jako parametr plik lub katalog nie istnieje.

2.7 Zadanie

Spróbuj uruchomić kolejną instancję powłoki i poleceniem `exit` zwrócić kod wyjścia 13. Sprawdź w powłoce wyżej czy otrzymano kod właśnie taki kod wyjścia.

2.8 Zadanie

Ustaw wartość zmiennej `JABLKA` na `2`. Skonstruuj polecenie uzupełniając wielokropki aby uzyskać wymagany tekst na wyjściu:

```
echo "Ja mam ...  jabłek.Ty masz ...  jabłek.On ma ...  jabłek."
```

ma wyświetlić

```
Ja mam 2 jabłek.Ty masz 4 jabłek.On ma 8 jabłek.
```

Musisz użyć zmiennej `JABLKA` oraz operacji matematycznych.

2.9 Zadanie

Dodaj do zmiennej środowiskowej `PATH` ścieżkę `$HOME/bin`.

2.10 Zadanie

Za pomocą operatorów obcinania napisów wyświetl tak obciętą ścieżkę `PATH` aby zawierała jedynie ścieżkę, którą należało dodać w poprzednim zadaniu.

2.11 Zadanie

Ustaw zmienną `TEKST` na wartość `"raz-dwa;trzy-cztery;piec-szesc;siedem-osiem"`. Obetnij zmienną `TEKST` tak aby uzyskać następujące napisy:

a. `raz-dwa`

b. `siedem-osiem`

c. `raz`

d. `osiem`

e. `szesc;siedem`

f. `dwa;trzy`

g. `szesc`

h. `piec`

i. `cztery;piec`

Uwaga: możesz używać zmiennych pośrednich jeśli potrzebujesz. Spróbuj nie używać liter we wzorcach.

2.12 Zadanie

Spróbuj wykonać powyższe ćwiczenie używając polecenia `cut` zamiast operatorów obcinania tekstu.