Ćwiczenie Nr 2
Konfiguracja środowiska użytkownika w systemie operacyjnym FreeBSD

Cel ćwiczenia:

Zapoznanie się ze środowiskiem pracy zwykłego oraz zaawansowanego użytkownika przy wykorzystaniu surowego systemu operacyjnego udostępnionego przez prowadzącego zajęcia. W ramach laboratorium przygotowany jest specjalny obraz maszyny z preinstalowanym systemem operacyjnym FreeBSD w minimalnej konfiguracji.

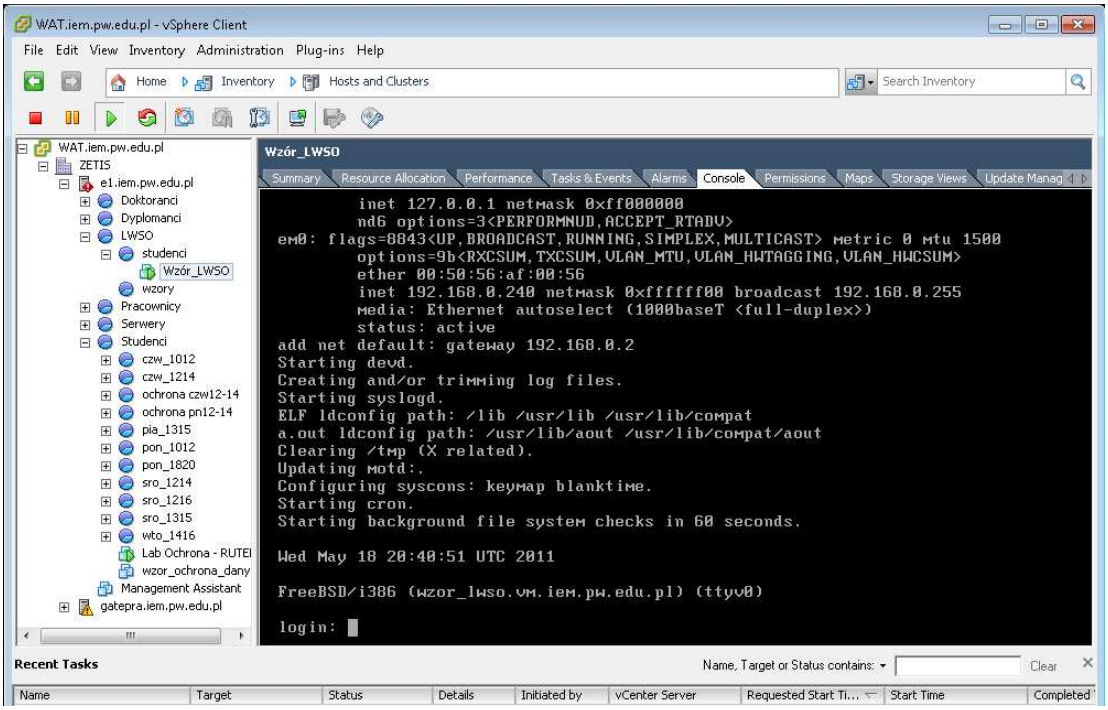
Celem ćwiczenia jest dokładne zapoznanie się ze środowiskiem Unixa, poznanie podstawowych opcji linii komend – powłoki, konfiguracja preferencji użytkownika. Uzyskiwanie i wyszukiwanie pomocy. Obsługa plików, praw, pliki ukrytych. Zarządzanie procesami użytkownika. Zapoznanie się ze strukturą drzewa katalogów. Zapoznanie się z technologią strumieni, i przetwarzania potokowego.

1. Scenariusz ogólny ćwiczenia

- Uruchomienie systemu z gotowego obrazu maszyny z systemem operacyjnym FreeBSD
- Zalogowanie do systemu i zapoznanie się konfiguracją sieciową
- Zapoznanie się z podstawowym edytorem vi
- Zarządzanie plikami i katalogami
- Konfiguracja zmiennych środowiska - shell, zmienne środowiskowe, aliasy, skrypty konfiguracyjne
- Odnajdowanie pomocy
- Zarządzanie uprawnieniami do plików
- Zarządzanie procesami
- Wykonywanie komend z wykorzystaniem przetwarzania potokowego
- Tworzenie skryptów w powłoce
- Kompilacja programów w języku C

2. Wykorzystanie udostępnionej maszyny wirtualnej

W celu realizacji poszczególnych elementów niniejszego ćwiczenia należy posłużyć udostępnioną przez prowadzącego maszyną wirtualną. Aby zalogować się do powłoki maszyny należy wykorzystać konto administratora maszyny root bez hasła. Należy w nazwie login podac Root,a gdy system zapyta o hasło wystarczy wcisnąć enter. Uzyskamy wówczas pełen dostęp do maszyny wirtualnej z uprawnieniami administratora. W trakcie niniejszego ćwiczenia należy samodzielnie przećwiczyć wszystkie komendy, które SA opisane w niniejszej instrukcji. Prawidłowe przećwiczenie jest niezbędne do realizacji przykładowych zdań na zaliczenie, które zostaną podane przez prowadzącego na zajęciach.



Rys. 2.1 Widok uruchomionej maszyny wirtualnej

3. Wprowadzenie teoretyczne i opis wykorzystywanych komend i narzędzi

Na początku ćwiczenia zapoznamy się z podstawowym edytorem tekstowym jakim jest **vi**. W narzędzie to są standardowo wyposażone wszystkie systemy Unixowe. Nazwa pochodzi od angielskiego skrótu vi – visual editor –edytor wizualny. W systemach Linuxowych dostępna jest rozszerzona wersja tego edytora pod nazwą vim – Enhanced vi.

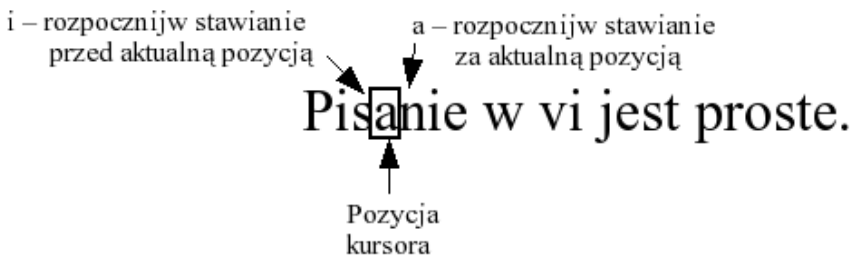
Vi umożliwia pracę w dwóch trybach: edycji i komend. Standardowo po uruchomieniu edytora pracujemy w trybie komend. Oznacza to, że możemy wykonywać komendy na dokumencie. Możemy kopiować, wklejać, usuwać, wyszukiwać, zapisywać itp. Nie możemy jednak tworzyć nowgo tekstu. Aby było trudniej, w vi mamy kilka trybów komend. Dla nas jednak wystarczy świadomość, dwóch trybów komend. Tryb komend klawiaturowych (domyślny w momencie uruchomienia) to po prostu wciśnięcia konkretnych klawiszy, np. klawisz h - powoduje przesunięcie kursora w lewo o jedną pozycję, klawisz b - powoduje przesunięcie kursora o jedno słowo w lewo, itp. Drugim trybem komend są komendy, które wpisujemy na ekranie (aby uruchomić ten tryb należy wcisnąć klawisz ':'

Aby rozpocząć dodawanie nowego tekstu do dokumentu, czyli po prostu pisać musimy przełączyć vi do trybu edycji. Sprawne przełączanie się między trybami edycji i komend jest podstawą w pracy z vi. Jest to jednocześnie jedna z najtrudniejszych czynności do przyswojenia sobie dla początkującego użytkownika, który przyzwyczajony jest, że każdy edytor otwiera się natychmiast w trybie edycji.

Przełączanie między trybami realizowane jest za pomocą następujących komend klawiaturowych:

- a - przejście do trybu edycji (rozpoczęcie dodawania nowego tekstu za znakiem aktualnie zasłoniętym przez kursor).
- i - przejście do trybu edycji (rozpoczęcie dodawania nowego tekstu przed znakiem aktualnie zasłoniętym przez kursor).

- r - przejście do trybu edycji tylko na jeden znak (Zastąpienie znaku aktualnie zasłoniętego przez kursor).
- R - przejście do trybu edycji (rozpoczęcie zastępowania wszystkich znaków).
- ESC - przejście do trybu komend.
- : - przejście do trybu komend wpisywanych na ekranie.



Rys.2.1 Schemat obrazujący przełączanie między trybami edycji i komend w edytorze vi

Utworzenie nowego dokumentu wpisanie jednego zdania, zapisanie zmian i wyjście:

```
volt# vi nowy.txt
```

Rozpoczęcie dodawania tekstu:

```
[Klawisz i]Pisanie w vi jest proste.[ESC][:][w][q]
```

- `[Klawisz i]` - ropoczęcie wstawiania
- `'Pisanie w vi jest proste.'` - wpisane zdanie
- `[ESC]` - powrót do trybu komend
- `[:]` - przejście do trybu komend wpisywanych na ekranie
- `[w]` - zapisanie zmian
- `[q]` - wyjście z vi

2.1 Niezbędne sekwencje komend vi dla każdego:

2.1.1 Jak wyjść z vi?

- [ESC]:q! - wyjście z vi bez zapisywania zmian
- [ESC]:wq - wyjście z vi z zapisaniem zmian (dokument musi mieć przyporządkowaną nazwę)

2.1.2 Jak zapisać dokument pod inną nazwą?

- [ESC]:w nowa_nazwa.txt - zapisanie dokumentu pod nową nazwą

2.1.3 Jak skopiować i wkleić fragment dokumentu?

- [ESC]yy - skopiowanie linii w której aktualnie znjaduje się kursor
- [ESC]dd - wycięcie linii w której aktualnie znjaduje się kursor
- [ESC]p - wklejenie skopiowanego tesktu za kursorem
- [ESC]P - wklejenie skopiowanego tekstu przed kursorem

Rys. 2.2 Schemat obrazujący kopiowanie wklejanie całych linii tekstu

2.1.4 Przykład obrazujący kopiowanie i wlejanie całych linii.

- [ESC]yw - skopiowanie znaków od kursora do końca słowa
- [ESC]x - usunięcie znaku zasłoniętego przez kursor
- [ESC]x$ - usunięcie tekstu od aktualnej pozycji kursora do końca linii.

2.1.5 Jak znaleźć słowo "lipa"?

- [ESC]/lipa[ENTER] - wyszukuje słowo "lipa", (komenda do szukania: /)
- [ESC]/[ENTER] - ponowne wyszukanie ostatniego słowa

2.1.6 Komendy przydatne podczas programowania.

- [ESC]:syntax on - włącza podświetlanie składni
- [ESC]:se ts=2 - ustawia głębokość wcięcia na 2
- [ESC]:se ai - włącza automatyczne wcięcie bloków programu

Wszystkie powyższe komendy można umieścić w pliku konfiguracyjnym vima: ~/.vimrc, który powinien wyglądać (znak ~ oznacza skrót do katalogu domowego):

```
syntax on
se ts=2
se ai
```

2.1.7 Pozostałe komendy:

- [ESC]:254 - przejdź do linii 254 w bieżącym pliku (komenda: 
- [ESC]]] - przejdź do następnej funkcji w pliku. (Funkcje muszą być sformatowane w ten sposób, że klamra otwierająca { musi być w następnej linijce co deklaracja funkcji.)
- [ESC][[- przejdź do poprzedniej funkcji w pliku

2.1.8 Automatyczne uzupełnianie wyrazów (tylko vim).

Automatyczne uzupełnianie wyrazów jest aktywne w trybie edycji. Aby automatycznie uzupełnić słowo: "huragan" wpisujemy np. początek słowa: "hu" a następnie wciskamy kombinację klawiszy: CTRL+P lub CTRL+N

- CTRL+P - próba dopasowania najbliższego słowa szukając wstecz dokumentu
- CTRL+N - próba dopasowania słowa przeszukując dokument do końca

Przykład:

Załóżmy że mamy, dokument i wciskamy we wskazanym miejscu [CTRL+P]:

Ala ma kota.
A_[CTRL+P]
Franek ma psa.

Spowoduje uzupełnie słowa "Ala" w miejscu wciśnięcia CTRL+P.

2.1.9 Tworzenie nowego pliku, uruchamianie vi:

a) vi nazwa.txt

b) w edytorze vi :

:w nazwa.txt

2.2 Zarządzanie katalogami:

md – make directory – utworzenie nowego katalogu,

mkdir nazwa – utworzenie nowego katalogu,

np.: mkdir –p ala/ma/kota

rmdir – usunięcie katalogu

rm –r [nazwa] – usunięcie katalogu wraz z plikami i podkatalogami (ostrożnie!)

2.3 Wyświetlanie zawartości pliku:

a) type, more, less, cat

- **cat nazwa.txt**

  podzielenie na ekrany | more

- **less nazwa.txt**

2.4 Usuwanie plików:

rm plik.txt – usunięcie pliku

rmdir katalog – kasowanie katalogu

rm –r – z podkatalogami

mv stara.txt nowa.txt – zmiana nazwy pliku

2.5 Powłoki tekstowe – konsole

Typy powłok:

sh – podstawowa powłoka

bash
ash
csh
zsh

} rozszerzenie sh

2.6 Podstawowe zmienne środowiskowe w sesji:

echo $PATH

echo $TEMP

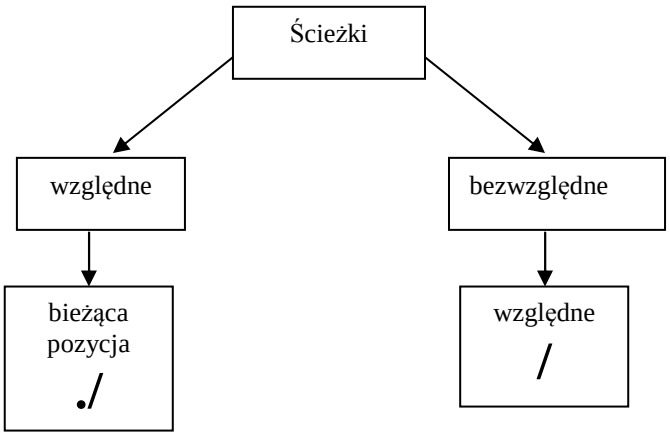
echo $HOME

echo $USER

PATH - Ścieżka systemowa – lista katalogów w którym dana nazwa programu w celu jego uruchomienia ma być odnaleziona.

PATH=/bin:/usr/bin:/usr/X1186/bin

W celu uruchomienia programu system przeszukuje zawsze zmienną systemową PATH. Cechą charakterystyczną systemu Unix jest to, że aby uruchomić program to musimy podać zawsze ścieżkę!



- - katalog bieżący

.. - katalog wyższy o 1 stopień

pwd – sprawdzenie bieżącej pozycji

W Unix jedno drzewo katalogów.

cd /

ls –la

2.7 Przykłady robienia aliasów

Aliasy są to skróty do często używanych komend.

alias lm=’ls -la’

alias a=’alias’

a k=’ls -l’

2.8 Podstawowe źródła pomocy

Manual – podręcznik

man printf – informacja o poleceniu printf

man 3 printf – 3 numer rozdziału pomocy polecenia printf

q – wyjście

whereis –wyszukiwanie pliku w systemie

locate – wyszukiwanie pliku w systemie z wykorzystaniem bazy indeksującej pliki

2.9 Zarządzanie uprawnieniami do plików i katalogów

UID – identyfikator użytkownika, właściciela

GID - identyfikator grupy, właściciela

Aby wyświetlić UID własnego użytkownika należy użyć polecenia:

id lub **whoami**

Typy uprawnień:

rwx

r – read

w – write

x – execute

atrybut praw – program, skrypt

user ID			prawa grupy			prawa wszystkich użytkowników		
r	**w**	**x**	**r**	**w**	**x**	**r**	**w**	**x**

Nadawanie praw dla pliku:

chmod

np. chmod 755 nazwa.txt

r	**w**	**x**	**r**	**-**	**x**	**r**	**-**	**x**
1	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**1**
7			**5**			**5**		

lub

np.: chmod u+r

u – user

g – group

o – other

a - all

chown – polecenie zmiany właściciela

np.:

chown szmurlor plik.txt

2.10 Zarządzanie procesami

Zarządzanie procesami – proces jest uruchomionym programem, czyli programem, który ma przydzielone zasoby pamięci operacyjnej, blok kontrolny procesu, identyfikator oraz znajduje się w kolejce procesów uruchomionych i co jakiś czas uzyskuje do własnej dyspozycji mikroprocesor:

- każdy proces ma strefę adresową
- każdy proces ma swój identyfikator

  PID – liczba całkowita typu integer

  W systemie zazwyczaj może być uruchomionych do 65385 procesów

  Proces ma zawsze właściciela (UID)

  Proces ma zawszę grupę będącą jego właścicielem (GID)

Aby wyświetlić listę procesów w systemie należy posłużyć się poleceniem:

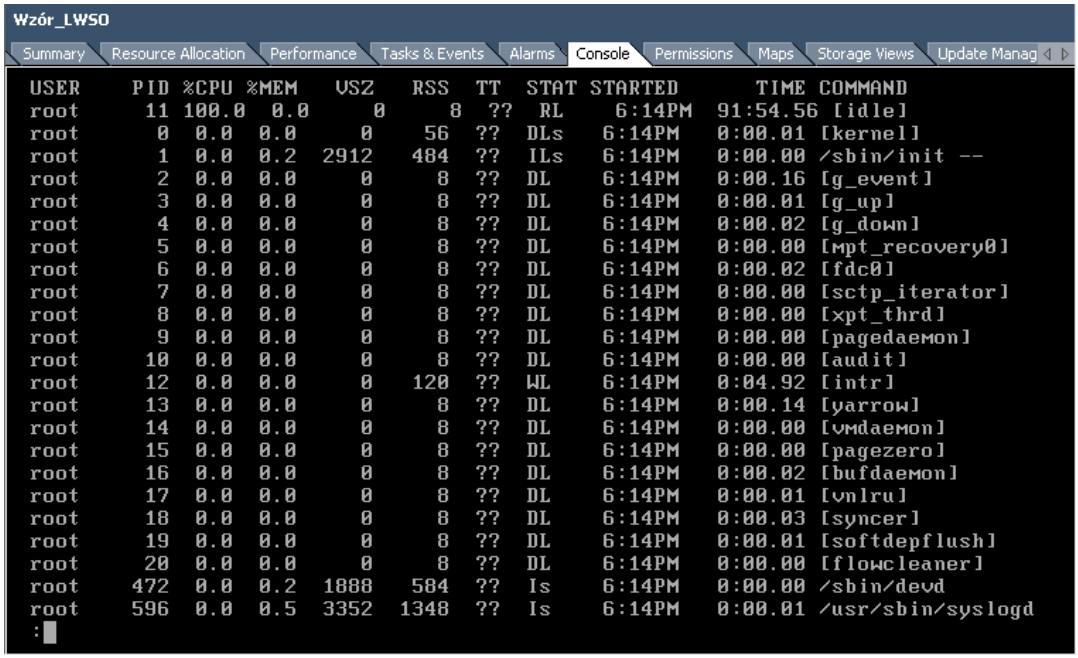
ps – lista procesów związanych z naszą sesją (naszym terminalem)

ps –aux – wyświetla wszystkie procesy uruchomione w systemie

Znacznie poszczególnych kolumn wyświetlanych przez komendę **ps**:

1	2	3	4	5	6	7	8	9	10	11
User	PID	aktualne użycie procesu	zużycie pamięci	rozmiar pamięci przydzielona dla całego procesu + biblioteki współdzielone	RSS pamięć fizyczna	Terminal TTY	STAT status procesu	czas uruchomienia	czas działania	proces z którym został uruchomiony

Przykład:



Bardzo interesująca jest kolumna ósma, w której znajduje się informacja o stanie procesu. Do najczęściej spotykanych stanów procesów należą:

8 – STAT:

R – proces uruchomiony (wykonuje działania)

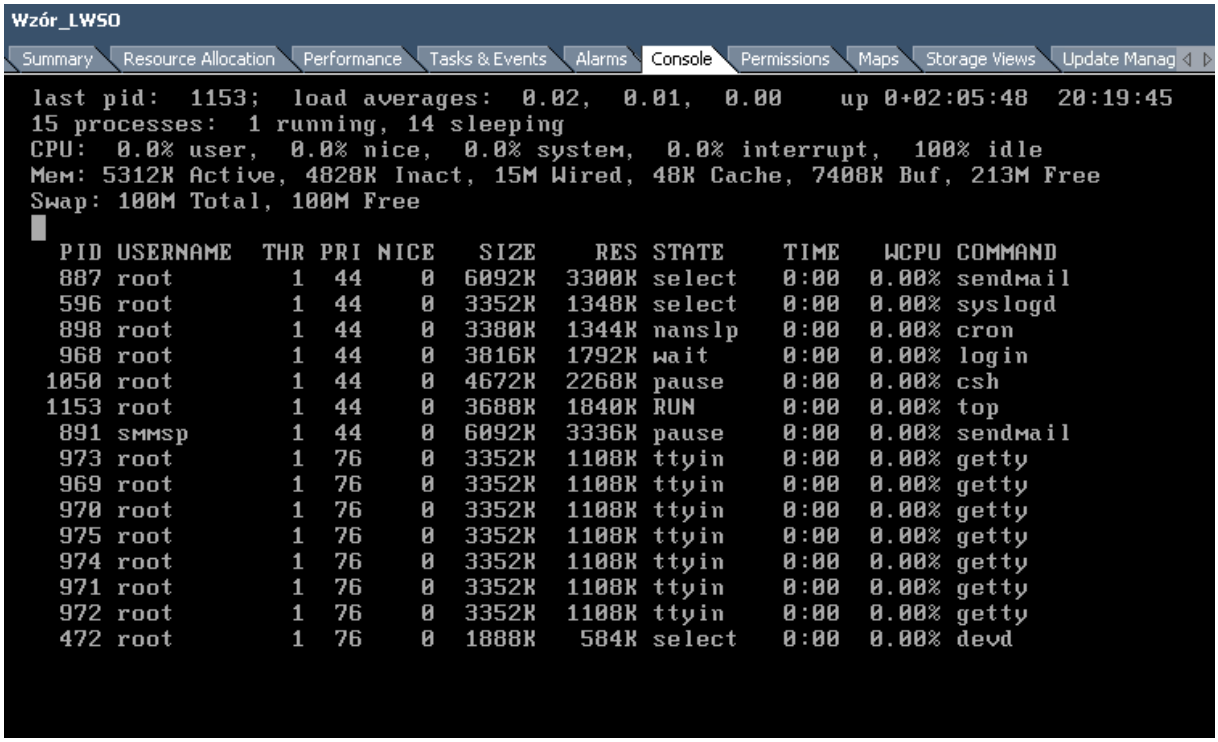
S – sleeping (uśpiony)

W – waiting (oczekuje na zdarzenie, np. połączenie sieciowe, lub wcisnięcie klawisza)

T – proces zawieszony

D – demon (proces uruchomiony w tle jako usługa)

Kolejną bardzo przydatną komendą wykorzystywaną do zarządzania procesami jest komenda **top**, która wyświetla, co jakiś czas odświeżając tablicę uruchomionych procesów posortowaną domyślnie po ilości użycia procesora. Oprócz informacji o uruchomionych procesach komenda ta zwraca również informacje o stanie systemu operacyjnego.



Operacje na procesach:

1) Ctrl+C – zatrzymanie procesu – jeżeli kontrolujemy jego klawiaturę

2) „Zabicie procesu” czyli przerwanie jego działania za pomocą komendy **kill -9 [PID]**, gdzie [PID] oznacza liczbę całkowitą oznaczającą identyfikator procesu.

3) Zawieszenie procesu. Posłużmy się przykładem. Uruchomijmy program top. Wciśnijmy CTRL+Z. System powinien wyświetlić komunikat „suspended”. Po czym przywróćmy działanie programu na wierzchu za pomocą komendy fg.

   Ctrl+Z – zawieszenie (zamknięcie)

   fg – przywrócenie funkcjonowania na wierzch sesji programu zawieszonego

   bg – proces działa w tle

2.11 Przetwarzanie strumieniowe - podstawy przetwarzania potokowego

W systemach operacyjnych występują następujące strumienie:

Nazwa	Skrót	Nr	Urządzenie
Standardowe wejście	STDIN	0	konsola (klawiatura)
Standardowe wyjście	STDOUT	1	konsola (monitor)
Standardowy kanał błędów	STDERR	2	konsola (monitor)

Standardowo powłoka czyta klawiaturę, a wyniki wykonanego programu oraz ew. błędy wykonania wyświetla na konsoli (czyli ekranie monitora) . Można to zmienić używając znaków **>**, **<**, **>>** .

Przykład 1. Standardowo polecenie ls wypisuje na konsolę (monitor) pliki i katalogi znajdujące się bieżącym katalogu; jego wejścia nie można zmieniać (ls zawsze czyta zawartość bieżącego katalogu), natomiast jego wyjście można skierować w inne miejsce niż do stdout

- przekierowanie wyjścia polecenia ls z konsoli (monitora) do pliku:

  ```
  ls > nazwa.txt
  ```

- chcąc dopisać zawartość do już istniejącego pliku piszemy:

  ```
  ls >> list.txt
  ```

Przykład 2. Polecenie cat czyta dane z stdin i wypisuje je na stdout .

- przekierowanie wejścia z konsoli (klawiatury) na plik (program nie będzie czytał klawiatury tylko przyjmie zawartość pliku):

  ```
  cat < pliki.txt
  ```

- przekierowanie zarówno stdout jak i stdin

  ```
  cat < plik1.txt > plik2.txt
  ```

- przekierowanie kanału błędu z konsoli do pliku:

  ```
  cat list2.txt 2> error.txt
  ```

- przekierowanie kanałów: wyjściowego i błędu do tego samego pliku:

  ```
  ls > wynik 2>&1
  ```

- można też podłączyć wyjście jednego programu do wejścia drugiego, np. polecenie:

  ```
  ls | sort
  ```

  wyświetli nam posortowana listę plików w danym katalogu
- potoki można stosować wielokrotnie; jeśli lista plików z poprzedniego polecenie jest zbyt długa, by się zmieścić na ekranie, można użyć polecenia:

  ```
  ls | sort | more
  ```

- potoki można rozwidlać: z jednego potoku tworzą się wówczas dwa, które można kierować do 2 różnych kanałów wyjściowych:

  ```
  who | sort | tee osoby.txt | more
  ```

  polecenie to spowoduje wypisanie identyfikatorów osób, pracujących w systemie (komenda who),

  wynik posortuje a następnie rozwidli potok i jeden z nich wyśle do pliku osoby.txt, a drugi wyświetli

  na konsoli komendą more

- potoki dają powłoce ogromne możliwości, odzwierciedlają też naczelną zasadę Unix'a: dużo prostych programów, które można łączyć w miarę potrzeb

- Przekierowanie standardowego wyjścia i wyjścia błędów do pliku:
- ps -aux ~/lista_procesow.txt 2>&1
- lub do urządzenia pustego czyli zignorowanie wszystkich komunikatów:
- ps -aux /dev/null 2>&1
- Przekierowanie stanardowego wyjścia i wyjścia błędów w różne miejsca:
- ps -aux 1>~/normalne_wyjscie.txt 2>~/bledy.txt
- Należy pamiętać, że znak > oznacza zapisanie do pliku z kasowaniem jego dotychczasowej zawartości natomiast znak >> oznacza dopisywanie do istniejącego już pliku.
- Polecenie kropka:
- . /etc/rc.d/init/http
- Polecenie to spowoduje dołączenie do aktualnie wykonywanego pliku pliku podanego jako parametr i natychmiastowe wykonanie go. Polecenie to można uznać za analogiczne do dyrektywy #include w języku C.

Przykład 3. Liczenie procesów.

Awk – jest to język skryptowy – wspomaga analizę tekstu.

ps –aux | awk ‘BEGIN {n=0;}{n=n+1;} END {print$h}’

Przykładowe zadania:

1. Podać maksymalny numer PID z listy aktualnie uruchomionych procesów. (Sprawdzić jak zmienia się numer nowo uruchamianego procesu.)

   Odp. ps –aux | awk ‘{print$2}’ | sort

2. Wyświetlić posortowaną listę wszystkich użytkowników którzy mają założone konto na serwerze volt.iem.pw.edu.pl. (Sprawdzić w pliku /etc/passwd.)

   Odp. cat /etc/passwd | cut –d ‘:’ –f1 | sort

3. Za pomocą komendy echo utworzyć plik o nazwie test_echo.txt z tekstem: 'Witaj to jest echo.

   Odp. echo „Witaj to jest echo” > test_echo.txt

2.12 Skrypty:

Aby nie powtarzać zestawów poleceń, często wydawanych powłoce, można zapisać je w pliku w postaci skryptu. Oto przykład banalnego skryptu:

```
#!\bin\bash
#
# Przykladowy skrypt
#
cd ~
```

```
echo "Jestesmy w katalogu domowym"
echo "Oto pliki w nim zawarte:"
ls -l
```

Celem jest przedstawienie ogólnych założeń programowania w powłoce. Niestety ze względu na krótki czas przeznaczony na tę część zajęć, celem nie jest nauczenie tworzenia własnych skryptów o ile umożliwienie zrozumienia skryptów systemowych jak i ich częściowe modyfikowanie.

Zacznijmy od podstaw składni. Linie rozpoczynające się od # stanowią komentarz. Pierwsza linia skryptu ma zazwyczaj następującą postać:

```
#!/bin/sh
```

Oznacza to umownie, że skrypt ma być interpretowany przez powłokę Bourne'a.

Prametry z linii komed w skryptach powłoki są przechowywane w postawci zmienncyh:

$1 – pierwszy parametrem

...

$9 – dziewiąty parametr

Aby mieć dostęp do pozstałych parametrów (więcej niż 9) należy użyć polecenia shift, które usuwa pierwszy parametr i przesuwa wszystkie w lewo. Innymi słowy parametr $1 ginie, a parametr $2 staje się parametrem $1 itd., aż do parametru $9, gdzie po wykonaniu pojedynczej komendy shift przechowany będzie dziesiąty parametr z linii komend.

$# - oznacza liczbę aprametrów (int argc z języka C).

“$@” równoważne jest zapisowi: “$1” “$2” .... “$n”.

$? - kod wyjściowy ostatnio wykonywanego polecenia.

2.13 Pozostałe przydatne właściwości powłoki tekstowej

Każde polecenie zwraca wartość . W systemach Unix jeżeli polecenie zakończyło się pomyślnie zwracana jest wartość 0. Gdy jakakolwiek część polecenia nie powiedzie się zwracana jest wartość niezerowa. W celu zwrócenia określonej wartości w skrypcie używa się polecenia exit:

np.

```
echo “blad przy konfiguracji drukarki”
exit(1)
```

Oznaczenia “||” oraz “&&” symbolizują potoki warunkowe.

Przykład:

```
grep robert /etc/passwd || echo “Brak uzytkownika robert w systemie.”
grep robert /etc/passwd || echo “Uzytkownik robert istnieje w systemie.”
```

2.14 Kompilacja programów w języku C:

W tej części ćwiczenia student powinien zdobyć podstawowe umiejętności kompilowania i uruchamiania programów w języku C. W tej części ćwiczenia będziemy wykorzystywać kompilator gcc. Rozważmy przykładowy program, który należy wprowadzić za pomocą omówionego wcześniej edytora vi do pliku pod nazwą **witaj.c**.

2.14.1 Przykład 1

Zawartość pliku witaj.c

```
#include <stdio.h>
int main(){
      printf(“Czesc\n”);
      return 0;
}
```

1.	Kompilacja:	gcc –c witaj.c
	wynik:	powstaje plik witaj.o
2.	konsolidacja:	gcc witaj.o
	wynik:	powstaje plik a.out
3.	konsolidacja:	gcc witaj.o –o witaj
	wynik:	powstaje plik wykonywalny pod nazwą witaj
4.	uruchomienie:	./witaj

2.14.2 Przykład 2

Należy utworzyć trzy pliki – z programem (witaj.c), nagłówkowy z deklaracją funkcji (bibliot.h) oraz z definicją funkcji (bibliot.c):

bibliot.c

```
void sumuj(int a, int b){
        printf(“%d+%d=%d\n”,a,b,a+b);
}
```

bibliot.h:

```
#ifndef_BIBLIOT
#define_BIBLIOT
void sumuj(int a, int b);
#endif
```

witaj.c:

```
#include <stdio.h>
int main(){
        printf(“Czesc\n”);
        sumuj(4,5);
}
```

Następnie należy skompilować pliki i uruchomić program wynikowy za pomocą komend:

1. gcc witaj.o
2. gcc –c bibliot.c
3. gcc witaj.o bibliot.o
4. gcc –c witaj.c

Aby kompilator dokładniej sprawdził poprawność napisanego programu oraz jego zgodność ze standardami warto jest wykorzystać specjalną opcję –Wall, która wyświetla zwiększoną ilość komunikatów ostrzegawczych.

5. gcc –c witaj.c – Wall - ostrzeżenie (warning)
6. gcc –c witaj.c -- pedantic

4. Przykładowe zadania i problemy na zaliczenie

1. Napisz skrypt w powłoce bash który wyświetli napis ‘TAK’ w sytuacji gdy istnieje plik o nazwie: /usr/local/vmware
2. Stworzyć plik witaj.sh postaci:

```
#!/bin/bash
# witaj.sh
# Prosty skrypt korzystajacy ze zmiennych shella.
echo "Witaj ${LOGNAME}!"
echo "Pracujesz na komputerze ${HOSTNAME}."
# opcja -n, aby nie bylo przejscia do nowej linii
echo -n "Dzisiejsza data: "
date
echo -n "Katalog biezacy: "
pwd
```

Nadajemy skryptowi prawo wykonywania poleceniem: chmod a+x witaj.sh

Uruchamiamy skrypt poleceniem: ./witaj.sh

3. Co to jest przetwarzanie potokowe. Podaj przykład komendy.
4. Wymień znane ci stany procesów procesów systemie unix.
5. Do czego służy komenda cut. Podaj przykład użycia.
6. Wymień 4 znane ci komendy które służą do operacji na procesach.
7. Co to jest proces zombie?
8. Jakie informacje możemy uzyskac za pomocą komend ifconfig i traceroute.
9. Przetłumacz na postać liczbową prawa: rwx- r-- --x, -w- --x r-x
10. Napisz skrypt w powłoce bash który wyświetli napis ‘TAK’ w sytuacji gdy istnieje plik o nazwie: /usr/local/vmware

11. Do czego służą i jak używać bg i fg.
12. Przetłumacz na postać tekstową prawa: 743, 365
13. Co to jest proces a co to jest program w sensie różnic?
14. Co znajduje się w katalogu /etc?
15. Co znajduje się w katalogu /proc?
16. Czym różnią się operatory: >>, >, |
17. Napisz przykład pliku makefile. Omów jego elementy.
18. Do czego służy komenda cut. Podaj przykład użycia.
19. Wymień 4 znane ci komendy które służą do operacji na procesach.
20. Co to jest proces zombie?
21. Jakie informacje możemy uzyskac za pomocą komend ifconfig i traceroute.
22. Jak w vi wyciąć całą linię i wkleić ją za linią aktualną po wycięciu.
23. Przetłumacz na postać liczbową prawa: rwx- r-- --x, -w- --x r-x

3. Literatura

1 L. J. Arthur, T. Burns, UNIX, Programowanie w Shellu, MIKOM 1998.

2 D. Taylor, 101 Skryptów w Shellu, MIKOM 2004.

3 Informacja o pakiecie sh-utils w systemie Linux (info sh-utils, man sh-utils).

4 L. Madeja, Ćwiczenia z systemu Linux, Korzystanie z pomocy, MIKOM, Warszawa 2000.

5 L. Madeja, Ćwiczenia z systemu Linux, Edytory vi, Emacs i sed, MIKOM, Warszawa 2001.