Zajęcia 1

- **uname** - wersja unixa
- **w, who, finger** - identyfikacja użytkownika
- **id, groups** - identyfikator i grupy
- **tty** - terminal
- **ssh, telnet** - zdalne logowanie
- **script** - zapisywanie terminala do pliku
- **quota** - przestrzeń dyskowa
- **exit, logout** - kończenie sesji / wylogowywanie
- **ftp, sftp** - transfer plików
    - *cd, lcd* - zmiana katalogu remote / local
    - *ls, lls* - lista plików remote / local
    - *get* - pobierz plik
    - *put* - wyślij plik
    - *bye* - zakończ sesję
- **scp** - kopiowanie plików (`scp [user@host:]plik_zródłowy [user@host:]plik_docelowy`)
- **wget** - pobieranie plików z sieci
- **mail** - maile

Zajęcia 2

- **ls (-a -l --ignore=)** - lista plików w katalogu
    - *-a* - wszystkie pliki
    - *-A* - wszystkie oprócz . i ..
    - *-l* - długi format
    - *-R* - rekursywnie wyświetla podkatalogi
- **mkdir (-p, -m)** - tworzenie katalogów
    - *-p* tworzy katalogi nadrzędne po drodze
    - *-m* ustawia chmode
- **rm** - usuwa pliki lub katalogi
    - *-f* - wymuszanie, ignoruje nieistniejące pliki, nie pyta o pozwolenie
    - *-R* - usuwanie rekursywne, katalog razem z zawartością
- **?** - zastępuje 1 znak
- ***** - zastępuje ciąg znaków o dowolnej długości
- Typy plików: TYPE: - d b c l p s
- **chmod**
    - chmod ugoa +-= rwx
    - chmod u+s,g+s
    - chmod 777
    - x - execute - wymagana żeby wykonać `cd` do katalogu
    - r - read - wymagana żeby wykonać `ls` w katalogu.
- **umask** - domyślne prawa dostępu
- **chown** - zmiana właściciela
- **chgrp** - zmiana grupy pliku
- **ln -s plik dowiązanie** - tworzenie linków symbolicznych
- `-rwxrwxrwx N uzytkownik grupa rozmiar data nazwa (N - dowiązania sztywne)`
- **mc** - midnight commander
- **tree** - struktura katalogu

Zajęcia 3

- **bash** - otwiera powłokę bash (Bourne Again Shell)
    - *zmienne*
      ZMIENNA=WARTOŚĆ
      echo $ZMIENNA (wyświetla wartość zmiennej)
      export ZMIENNA=WARTOŚĆ (eksportuje do innych sesji)
      unset ZMIENNA (usuwa zmienną)

    - *zmiana znaku zachęty*
      export PS1='\u@\h:\w '

    - *pliki*
      .**bash_profile** / **.bash_login** / **.profile** - uruchamiane dla powłoki login
      .bashrc - uruchamiane dla powłoki non-login
      .bash_logout - uruchamiane przy zamykaniu powłoki

- **tcsh** - otwiera powłokę (C shell)
    - *zmienne*
      setenv ZMIENNA WARTOŚĆ
      printenv ZMIENNA (wyświetla wartość zmiennej)
      unsetenv ZMIENNA WARTOŚĆ

    - *zmiana znaku zachęty*
      set prompt = "%n@%m:%~ "

    - *pliki*
      .**login** - uruchamiane dla powłoki login
      .tcshrc - uruchamiane dla każdej powłoki
      .logout - uruchamiane przy zamykaniu powłoki
- **alias** alias_polecenia=’polecenia’ - ustawia alias dla polecenia
- **unalias** alias_polecenia - usuwa alias
- **“ … ”** - **cudzysłowy** - tekst, w którym powłoka może interpretować znaki specjalne, np. podstawiać wartości zmiennych
- **‘ … ’** - **apostrofy** - j.w. tylko BEZ możliwości interpretowania znaków specjalnych - tekst jest dokładnie cytowany
- **` … `** - **backtick** - odwrócone apostrofy - zawartość tekstu jest traktowana jako polecenie, uruchamiana przez kolejna powłokę, a wynik działania polecenia jest podstawiany jako tekst.
- **vi** / **vim** / **emacs** - edytory tekstu dla masochistów
- **less** / **more** - do przeglądania długich tekstów
- **cat** [plik] - wyświetla zawartość pliku

Zajęcia 4

Wypisywanie całości plików

- **cat** łączenie i wypisywanie plików
- **tac** łączenie i wypisywanie odwróconych plików
- **nl** numerowanie linii i wypisywanie plików
- **od** wypisywanie plików w formacie ósemkowym i innych

Formatowanie zawartości plików

- **fmt** reformatowanie akapitów tekstu
- **pr** stronicowanie i kolumnowanie plików do wydruku
- **fold** zawijanie linii wejściowych do zadanej szerokości

Wypisywanie części plików

- **head** wypisywanie początku plików
- **tail** wypisywanie końcówki plików
- **split** podział pliku na części stałej wielkości
- **csplit** podział pliku na części zależne od kontekstu

Podsumowywanie plików

- **wc** wypisywanie liczby bajtów, słów i linii
- **sum** wypisywanie sumy kontrolnej i liczby bloków
- **csum** wypisywanie sumy CRC liczby bloków
- **md5sum** wypisywanie lub sprawdzanie skrótu danych

Sortowanie i działania na plikach posortowanych

- **sort** sortowanie plików tekstowych
- **uniq** pozostawianie unikalnych linii w pliku
- **comm** porównywanie dwu posortowanych plików liniami
- **ptx** tworzenie indeksu permutacyjnego zawartości pliku
- **tsort** sortowanie topologiczne

Działania na polach wewnątrz linii

- **cut** wypisywanie wybranych części linii
- **paste** zlepianie linii plików
- **join** łączenie linii według wspólnego pola

Działania na znakach

- **tr** zamiana, ściskanie, usuwanie znaków
- **expand** zamiana tabulacji na spacje
- **unexpand** zamiana spacji na tabulacje

GREP

grep [opcje] wzorzec [plik]

- -A x – pokazuje x linii po znalezionym wzorcu
- -B x – pokazuje x linii przed znalezionym wzorcem
- -C x - pokazuje x liniii przed i po znalezionym wzorcu
- -x – zaznacza tylko te dopasowania, które pasują do całej linii
- -i – "znieczula" grep'a na wielkość liter. W tym przypadku wyrazy Linux i LInuX będą traktowane jak identyczne.

- -E – używa rozszerzonych wyrażeń regularnych

WYRAŻENIA REGULARNE - PODSTAWOWE

- . dowolny znak
- ^ poczatek linii, np. ^A linia zaczynajaca sie od A
- $ koniec linii
- [abc] a lub b lub c
- [a-zA-Z] dowolna litera
- [^0-9] dowolny znak procz cyfr ([^ dopelnienie zakresu)
- ^ab$ linia ab
- .*ok.* dowolny napis zawierajacy "ok" (takze samo "ok")
- [-^] myslnik lub daszek
- [^^] dowolny znak oprocz "^"

Zajęcia 5

1. Znaleźć przy pomocy find:

- **w katalogu /usr/bin wszystkie pliki o nazwie zaczynającej się na au,**
  `find /usr/bin -name “au*”`

- **w katalogu /tmp wszystkie pliki należące do użytkownika root,**
  `find /tmp -user root`

- **w katalogu /tmp wszystkie pliki modyfikowane w ciągu ostatnich 24 godzin**
  `find /tmp -mtime -1`

  Wytłumaczenie:
  W manualu przeczytamy: `-a/c/mtime n*24`
  oznacza to, że stosując parametr -a/c/mtime określamy ile 24-godzinnych okresów temu plik miał być odpowiednio:

  `-atime` => kiedy ostatnio dostano się do pliku (a = accessed)
  `-ctime` => kiedy ostatnio zmienił się status pliku (c = changed)
  `-mtime` => kiedy ostatnio modyfikowano dane pliku (m = modified)

  argument n możemy zdefniować np.

  +1 => zmiana nastąpiła więcej niż 24 godziny temu *(>24*1) (od prehistorii do 24 godzin od teraz)*
  1 => zmiana nastąpiła dokładnie 24 godziny temu *(=24*1)*
  -1 => zmiana nastąpiła mniej niż 24 godziny temu (<24*1) *(od 24 godziny temu do teraz)*

  a zatem żeby sprawdzić jakie pliki zostały zmodyfikowane w ciągu ostatnich 24 godzin chcemy znaleźć pliki, które zostały zmienione “od 24 godzin temu do teraz”, stąd `-mtime -1`

  `Dodatkowo warto pamiętać o -a/c/mmin który działa podobnie ale zamiast 24 godzinnych okresów przyjmuje czas w minutach`

- **w katalogu /tmp wszystkie pliki zwykłe o uprawnieniach 700.**
  `find /tmp -perm 700`

- **w katalogu /tmp wszystkie pliki o nazwie “trywialne” bez względu na wielkość znaków**
  `find /tmp -iname “trywialne”`

- **w katalogach /tmp, /usr/bin, /home wszystkie pliki zawierające “test”**
  `find /tmp /usr/bin /home -name “*test*”`

- **w katalogu /tmp wszystkie pliki o rozszerzeniu “.txt” lub “.deb”**
  `find /tmp -name “*.txt” -o -name “*.deb”`
  (-o działa jak operator logiczny OR)

- **w katalogu domowym znajdź wszystkie pliki zawierające “file” i ustaw im chmod 700**
  `find ~ -name “*file*” -exec chmod 700 {} \;`

- **w katalogu domowym znajdź wszystkie pliki będące katalogami i wyświetl jak polecenie ls**
  `find ~ -type d -ls`

- **znajdzie tylko pliki tylko w katalogu /usr/bin (nie w podkatalogach)**
  `find /usr/bin -maxdepth 1 -type f`

- **znajdzie w katalogu /usr/bin wszystkie pliki, których nazwa nie zaczyna się od “a”**
  `find /usr/bin -not -name “a*”`

2. Zaawansowane filtry - napisać filtr, który:

- **wyświetli 7 pierwszych, posortowanych alfabetycznie, nazw plików o nazwie kończącej się na p znajdujących się w katalogu /usr/bin.**
  `find /usr/bin -name “*p” | sort | head -n 7`

- **poda liczbę wszystkich plików zawierających w nazwie słowo ubuntu, znajdujących sie w katalogu /usr/bin.**
  `find /usr/bin -name “*ubuntu*” | wc -l`

- **zapisze do pliku spis posortowaną alfabetycznie listę użytkowników aktualnie zalogowanych z komputerów mających początek adresu IP: 192.168.0 (inna opcja to w przypadku wyświetlania nazw domen: w domenie .agh.edu.pl).**
  `who | awk ‘/.*192.168.0.*/ {print $1}’ | sort >> spis`
  --- lub ---
  `who | grep “.*192.168.0.*” | awk ‘{print $1}’ | sort >> spis`

- **wybierze z 6 ostatnich linii pliku spis te, które zawierają słowo cos i poda ilość występujących w nich znaków. (plik testowy należy stworzyć).**
  `tail -n 6 spis | grep “.*cos.*” | wc --chars`

3. Tworzenie skryptów - napisać poprawny skrypt, który wyświetli:

Uwaga do skryptów

```
$(polecenie) = `polecenie`
```

Notacja $(...) działa tylko w powłoce bash, co znaczy, że gdyby skrypt zaczynał się #!/bin/sh to należy używać backticków ``. W powłoce bash działają obydwie notacje i można ich używać zamiennie.
Dla dociekliwych: http://mywiki.wooledge.org/BashFAQ/082

Aktualną datę, nazwę i czas pracy systemu

```
#!/bin/bash
TIME=$(uptime | awk -F , '{print $1}')

echo "Czas systemu: ${TIME}"
echo "Nazwa systemu: $(uname)"
```

Spis zalogowanych użytkowników i spis sesji osoby uruchamiającej skrypt

```
#!/bin/bash

echo "Zalogowani użytkownicy: $(users)"
echo "Spis sesji: $(cat ${HOME}/.bash_history)"
```

Wyświetli prawa dostępu do pliku podanego jako parametr z linii poleceń: skrypt plik i poda typ/opis tego pliku. Do powyższego skryptu dopisać (najlepiej w postaci funkcji) kod sprawdzający czy właściciel pliku jest zalogowany, a jeżeli tak to wyświetli informacje o nim i jego otwarte sesje.

```
#!/bin/bash

function print_owner_info() {
  if $(users | grep "$1" -q); then
    echo "Wlasciciel pliku jest aktualnie zalogowany."
    echo "Informacje o wlascicielu pliku:"
    w "$1"
  fi
}

ACCESS=$(stat --format="%a" $1)
TYPE=$(stat --format="%F" $1)
OWNER=$(stat --format="%U" $1)

echo "Prawa dostepu: ${ACCESS}"
echo "Typ pliku: ${TYPE}"
print_owner_info $OWNER
```

Napisać skrypt (np. do użycia jako CGI) generujący dynamicznie stronę WWW (plik HTML) o tytule: To jest spis plików w katalogu KATALOG zawierającą spis i opis plików w podanym z linii poleceń

katalogu, na każda pozycja spisu jest hiperłączem do odpowiedniego pliku.

```
#!/bin/bash

function generate_link() {
  URL=$1
  NAME=$(basename $1)
  echo "<li><a href=\"${URL}\">${NAME}</a></li>"
}

HTML="<html><head><title>To jest spis plików w katalogu
${1}</title></head><body><ul>"
HTML_LINKS=""

FILES=$(find $1 -maxdepth 1 -type f | tr "\n" " ")
echo $FILES

for FILE in $FILES; do
  HTML_LINKS="${HTML_LINKS} $(generate_link $FILE)"
done

HTML="${HTML} ${HTML_LINKS} </ul></body></html>"

echo $HTML > ./list.html
```

--- PISANIE SKRYPTÓW: WSKAZÓWKI ---

1. Instrukcje warunkowe

```
# Tutaj if oczekuje juz gotowej wartosci logicznej (true/false)
if warunek; then
  ...
fi

# Tutaj if uruchamia program test (wpisz man [ to sie zdziwisz)
# W srodku [] wykonywane jest porownanie i zwracana jest wartosc logiczna
if [ warunek do zbadania ]; then
  ...
fi

# Podwojne nawiasy kwadratowe rozszerzaja dostepne nam polecenia testowe
# Mozemy np. teraz korzystac z:
#   - pattern matchingu: [[ $name = a* ]]
#   - regexpa [[ $(date) =~ ^a.* ]]
#   - operatorow logicznych: && (and), || (or)
# Wiecej info: http://mywiki.wooledge.org/BashFAQ/031
if [[ warunek do zbadania ]]; then
  ...
fi

# Jezeli chcemy sie posluzyc grepem w if'ie to korzystamy
# z opcji -q (quiet). Jezeli pattern zostanie odnaleziony,
# to instrukcja w if'ie sie wykona
if grep -q "^test$" ./file.txt; then
  ...
fi
```

2. Pętle

```
# Petla przechodzi przez wszystkie elementy oddzielone spacja
for N in 1 2 3 4 5 6 7 8 9; do
  echo $N
done

# To samo co wyzej tyle ze generujemy
# liczby od 1-9 przy uzyciu brace expansion
for N in {1..9}; do
  echo $N
done

# Loopowanie po liniach pliku
while read -r LINE; do
  echo $LINE
done < /etc/passwd
```

3. Operacje arytmetyczne

```
#!/bin/bash

# Kiedy chcemy przypisac do zmiennej wartosc
# jakiejs operacji arytmetycznej, poslugujemy sie
# (podobnie jak w przypadku polecen) notacja $(), tyle ze
# tym razem stosujemy podwojne nawiasy:
A=$((2 + 2))
B=$((3 + 3))

echo "${A}, ${B}"

# Kiedy odwolujemy sie do zmiennych w srodku operacji
# arytmetycznej, mozemy poprzedzic ich nazwy $, ale nie musimy
C=$((A * 2 + B))
D=$(($A * 2 + $B))
echo "${C}, ${D}"

# Inkrementacja zmiennych:
# Kiedy inkrementujemy zmienne, nie poprzedzamy
# nawiasow ani nazwy zmiennej $
(( A++ ))
```

```bash
# Reszta z dzielenia (modulo)
echo $(( A % 2 ))

# UWAGA:
# Powloka bash nie obsluguje operacji na liczbach
# zmiennoprzecinkowych. W tym celu musimy wykorzystac
# inne polecenia dostepne w systemie Unix

# Wypisuje 0
echo $((2 / 3))

# Wypisuje .6666666666666
#   bc => basic calculator
#   bc -l => uruchamia basic calculator i wczytuje math library
#            Przelacznik wymagany do operacji na zmiennoprzecinkowcyh
echo $(bc -l <<< "2/3")

# Wypisuje .666667
awk '{print $1 / $2}' <<< "2 3"

# Porownywanie liczb w if'ach
#   -eq | A == B | (equal)
#   -ne | A != B | (not equal)
#   -ge | A >= B | (greater than or equal)
#   -gt | A >  B | (greater than)
#   -le | A <= B | (less than or equal)
#   -lt | A <  B | (less than)
if [ $A -gt $B ]; then
  echo "Super"
fi

# Porownywanie liczb zmiennoprzecinkowych
#   Wytlumaczenie:
#   Przekazanie rownania za pomoca pipe do bc zwraca 1 lub 0
#   Jezeli opakujemy ta wartosc w podwojne nawiasy (( )) zostanie
#   ona przetlumaczona na wartosc logiczna true / false
if (( $(echo "2/3 > 1/3" | bc -l) )); then
  echo "Super float"
```

```
fi
```

4. Operacje na zmiennych tekstowych

```
#!/bin/bash

STRING="abcdefabc"

# Pobieranie dlugosci zmiennej
# ${#NAZWA_ZMIENNEJ}
LENGTH=${#STRING}

echo "${LENGTH}"

# Usuwanie podciagu z ciagu znakow
# (Usuwa tylko pierwsze wystapienie)
echo ${STRING#"abc"}

# Podmiana znakow w ciagu znakow
# (Podmienia tylko pierwsze wystapienie)
echo ${STRING/"abc"/"123"}

# Podmiana znakow w ciagu znakow
# (Podmienia wszystkie wystapienia)
echo ${STRING//"abc"/"123"}

# Wyciaga z ciagu znakow okreslony podciag
#   ${ZMIENNA:start:dlugosc}
#   - pozycje znakow liczone od zera ("abc" - a = 0, b = 1, c = 2, itd.)
#   - jezeli nie podamy dlugosci to pobiera od startu do konca stringa
#   - jezeli podamy tylko dlugosc to pobiera od poczatku do danej dlugosci

# Wyswietli "de"
echo ${STRING:3:2}

# Wyswietli "defabc"
echo ${STRING:3}

# Wyswietli "ab"
echo ${STRING::2}
```

Zajęcia 6

------ SED ------

1. **Wyświetlić plik /etc/passwd przy pomocy *sed*.**
   ```
   sed '' /etc/passwd
   ```

2. **Zamienić separator - dwukropek - w pliku /etc/passwd na spację.**
   ```
   sed 's/:/ /g' /etc/passwd
   ```

3. **Wyświetlić tylko loginy użytkowników zapisanych w pliku /etc/passwd**
   ```
   sed 's/:.*//' /etc/passwd
   ```

4. **Wyświetlić 4, 7, 10 i 13 linię pliku /etc/passwd**
   ```
   sed -n '4~3 p; 13 q' /etc/passwd
   ```

5. **Wyświetlić tylko 5 linię**
   ```
   sed '5!d' file.txt
   ```

6. **Wyświetlić określone przedziałem (np. od 3. do 5. włącznie) linie pliku /etc/passwd.**
   ```
   sed -n '3,5 p' /etc/passwd
   ```

7. **Wyświetlić linie pliku /etc/passwd opisujące osoby mające login zaczynający się na 'z'.**
   ```
   sed -n '/^z/p' /etc/passwd
   ```

8. **Wyświetlić linie pliku /etc/passwd opisujące osoby mające login zaczynający się na 'w' lub 'z'.**
   ```
   sed -n '/^[wz]/p' /etc/passwd
   ```

9. **Jak przy pomocy sed zaimitować polecenie grep -v? np. dla frazy 'lo' (grep -v lo /etc/networks)**
   ```
   sed -n '/lo/!p' /etc/networks
   ```

10. **Jak zamienić w pliku wszystkie słowa root na twój login (przetestuj na pliku /etc/aliases)?**
    ```
    sed 's/root/login/g' /etc/aliases
    ```

11. **Jak zamienić słowo 'root' na twój login w pliku, ale tylko w wierszach, w których występuje 'www'? A jak tam gdzie nie występuje?**
    ```
    sed '/www/ s/root/login/' file.txt
    sed '/www/! s/root/login' file.txt
    ```

12. **Jak usunąć z pliku puste linie?**
    ```
    sed '/^$/d' file.txt
    ```

13. **Jak zamienić przy pomocy sed wszystkie litery 'r' na 'k'?**
    ```
    sed 's/r/k/g' file.txt
    ```

14. **W jaki sposób zakodować szyfrem ROT13 plik przy pomocy sed (szyfr zamienia litery na występujące 13 liter dalej, np. a↔n, b↔o, itd.)?**
    ```
    sed
    'y/abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ/nopqrstuvwxyzabcdefg
    hijklmNOPQRSTUVWXYZABCDEFGHIJKLM/' file.txt
    ```

15. **Przy pomocy polecenia sed zakomentuj linijkę link-local w pliku /etc/networks.**
    ```
    sed '/^link/ s/^/# /' /etc/networks
    ```

16. **W jaki sposób przy użyciu sed wstawić kolumnę X po pierwszym znaku wiersza (dodatkowy znak X w każdym wierszu)? A jak po piątym?**
    ```
    sed 's/^\(.\)/\1X/' /etc/networks
    ```

17. **Jak przy pomocy sed powtórzyć 3 razy pierwsze dwie litery każdego wiersza w pliku?**
    ```
    sed -n 's/^\(.\)\(.\)\(.*\)/\1\1\1 \2\2\2/p' /etc/networks
    ```

18. **Wylistuj wszystkie wiersze pliku /etc/mime.types zaczynające się od video i wyświetl ich numer.**
    ```
    sed '/^video/{p;=}' /etc/mime.types
    ```

19. **Napisz polecenie sed imitujące polecenie cut -d: -f2.**
    ```
    sed 's/\(^[^:]*\):\([^:]*\).*/\2/' /etc/passwd
    ```

20. W jaki sposób zmienić kolejność słów (np. w pliku /etc/aliases)? (Jest: news: root -> Zrób: root: news)
21. **Napisz polecenie sed imitujące cat -n.**
    ```
    sed = /etc/passwd | sed 'N;s/\n/\t/'
    ```

22. **(+)** Napisać **skrypt** programu *sed* który ustawi powłokę startową wszystkim użytkownikom grupy *is1* (folder domowy w katalogu *is1*) na /bin/tcsh. Skrypt ma wydrukować całą zawartość zmienionego pliku na ekranie wraz z zaznaczeniem zmienionych linii - tak jak jest to przedstawione poniżej.
    ---
    linia w której nastąpiła zmiana powłoki

    ```
    sed 's/\(.*\):\(*\):\(*\):\(*\):\(*\):\(*\):\(.*\/is1\/*\):\(*\)/------\n
    \1:\2:\3:\4:\5:\6:\/bin\/tcsh\n-------'
    ```

23. Napisać **skrypt** programu *sed* wyświetlający linię zawarte w pliku /etc/passwd w odwrotnej kolejności.

--- AWK ---
Jak awk interpretuje spacje? - [[:space:]]

Co określają następujące zmienne: FS, RS, NF, NR, OFS, ORS?

- FILENAME : current input file name(do not change variable name)
- FR : number of the current input line (that is input line 1, 2, 3… so on, do not change variable name)
- NF : number of fields in current input line (do not change variable name)
- OFS : output field separator
- FS : input field separator
- ORS : output record separator
- RS : input record separator

Zajęcia 7

Ćwiczenie 1. Słownik krzyżówkowy

- Proszę napisać skrypt, który dla słowa podanego jako argument, w którym dowolne znaki wstawiamy jako “.”, wyświetli znalezione dopasowania ze słownika.
  np. dla: ./skrypt b.eb.. znalezionym dopasowaniem mogłoby być bieber.
- W przypadku, gdy nie podamy słowa, skrypt powinien wypisać komunikat “Nie podano słowa”.
- W przypadku, gdy nie podamy słowa, skrypt powinien ponadto poprosić użytkownika o wprowadzenie słowa i pobrać od niego to słowo.
- Skrypt powinien sprawdzać, czy pobrane słowo nie zawiera niedozwolonych znaków (innych niż litery i kropki) i wyświetlić komunikat „Podano nieprawidłowe znaki”.

```
#!/bin/bash

QUERY=$1

if [ -z $QUERY ]; then
  read -p "Nie podano hasla do wyszukania. Podaj je teraz: " QUERY
fi

if [[ "$QUERY" =~ [^a-zA-Z.] ]]; then
  echo "Podano niedozwolone znaki"
  exit
fi

grep "${QUERY}" /tmp/dictionaryen.txt
```

Ćwiczenie 2. Deszyfrator szyfru przestawieniowego

```
#!/bin/bash

LETTERS="abcdefghijklmnopqrstuvwxyz"
LENGTH=${#LETTERS}
MATCH_TRESHOLD=0.3

#SEARCH='wrqan m plse cvah gb fvrqrz'
SEARCH='unjwbxej heyjwd hdkwd unsz yfpnj xfrj f ptqjosj yt uwejxzsnęhnj qnyjw b
yjo bnfitrtśhn'

for I in $(seq 1 $LENGTH);
do
  WORD_COUNT=0
  WORDS_FOUND=0
  DECODED=$(sed "y/${LETTERS}/${LETTERS:$I}${LETTERS::$I}/" <<< $SEARCH)
  echo "Przesuniecie o ${I} znakow: ${DECODED}"

  DECODED_ARRAY=$(tr " " "\n" <<< $DECODED)

  for WORD in $DECODED_ARRAY;
  do
    ((WORD_COUNT++))
    if grep -q "^${WORD}$" /tmp/slownikpl.txt;
    then
      echo "  -> Znaleziono słowo '${WORD}' w słowniku"
      ((WORDS_FOUND++))
    fi
  done

  if (( $(echo "(${WORDS_FOUND}/${WORD_COUNT})>${MATCH_TRESHOLD}" | bc -l) ));
  then
    echo -e "    + Ciąg \"${DECODED}\" jest zgodny ze słownikiem."
    echo -e "    + Przesunięcie w tym szyfrze wynosi ${I}."
  fi
done
```

Ćwiczenie 3. Kreator krzyżówek

```
#!/bin/bash

# Losowa liczba z zakresu [a, b]
seq 1 10 | shuf -n 1

# Losowa linijka z pliku
shuf -n 1 /tmp/krzyzowkowe.txt

# Wypisz tylko hasla bez opisow, wybierz losowo jedno
sed -n '1~2 p' /tmp/krzyzowkowe.txt | shuf -n 1

# Wypisz tylko hasla bez opisow, nastepnie wybierz tylko te,
# ktore zaczynaja sie na litere b, po czym wybierz jedno losowo
sed -n '1~2 {/^b/p}' /tmp/krzyzowkowe.txt | shuf -n 1
```

Ćwiczenie 4. Lista mailingowa

```
#!/bin/bash

if [ $1 = "-a" -o $1 = "--add" ]; then
  if grep -q "${2}" listaodbiorcow; then
    echo "${2} jest juz na liscie odbiorcow!"
    exit
  fi

  if grep -vq '^[a-zA-Z0-9]\+@[a-zA-Z0-9]\+\.[a-z]\{2,\}' <<< $2; then
    echo "Podaj poprawny adres e-mail!"
    exit
  fi

  echo $2 >> listaodbiorcow
  echo "Dodano ${2} do listy odbiorcow"
  exit
fi

if [ $1 = '-r' -o $1 = "--remove" ]; then
  # Przelacznik -i w sedzie sprawia ze zapisuje zmiany do pliku
  sed -i "/${2}/d" listaodbiorcow
  echo "Usunieto ${2} z listy odbiorcow"
```

```
  exit
fi

read -p "Podaj temat maila: " SUBJECT
read -p "Podaj tresc maila: " MESSAGE

TO=$(cat listaodbiorcow | tr "\n" " ")

echo $MESSAGE | mail -s $SUBJECT $TO
```

Zajęcia 8

1. GPG

1. Wygenerować swój klucz
`gpg --gen-key`

2. Zaszyfrować i rozszyfrować wybrany plik dla siebie,
`gpg --encrypt --armor -o <plik_wynikowy> <wiadomosc>`
`gpg --decrypt -o <plik_wynikowy> <zaszyfrowana_wiadomosc>`

3. Wyświetlić listę kluczy:
`gpg --list-keys`
UWAGA: identyfikator klucza, to dowolny fragment linii z napisem (a nie login!)

4. Eksport klucza:
`gpg -a --export [id_klucza] > [nazwa_pliku]`

5. Import klucza:
`gpg --import <plik_klucza>`

6. Podpisać elektronicznie plik,
`gpg --clearsig -o <plik_wynikowy> <wiadomosc>`

2. SSH

1. Generowanie klucza SSH
`ssh-keygen`

3. Hash

1. Wyliczyć przy pomocy md5sum skrót wybranego pliku.
`md5sum <plik>`

Zajęcia 9

1. Założyć konto przy pomocy polecenia useradd

- **wywołując je tylko z nazwą użytkownika**
  `useradd user | adduser user`

- **oprócz nazwy użytkownika ustawić z linii poleceń powłokę**
  `useradd -s /bin/sh <username> | adduser --shell /bin/sh <username>`

- **oprócz nazwy użytkownika ustawić z linii poleceń katalog domowy**
  `useradd -d /home/katalog -m <username> | adduser --home <directory> <username>`

- **oprócz nazwy użytkownika ustawić z linii poleceń numer uid**
  `useradd -u 999 <username> | adduser --uid <uid> <username>`

- **oprócz nazwy użytkownika ustawić z linii poleceń grupę użytkownika**
  `useradd -g grupa <username> | adduser --group <groupname> <username>`

- **Czy padło pytanie o hasło ? Jeżeli nie, należy je ustawić poleceniam passwd.**
  `passwd <username> | W przypadku adduser zawsze jesteśmy pytani o hasło`

2. Sprawdzić przy pomocy useradd -D domyślne parametry konta, a następnie zmienić:

- **domyślną powłokę**
  `useradd -D -s /bin/sh`

- **domyślny katalog domowy**
  `useradd -D -b /home/katalog`

- **domyślną grupę**
  `useradd -D -g grupa`

- **wykonanie powyższych poleceń dla polecenia adduser**
  `Edytujemy odpowiadające linijki pliku /etc/adduser.conf`

```
#Domyślna powłoka
DSHELL=<powłoka>
#Katalog Domowy
DHOME=<katalog domowy>
#Grupa domyślna, jeżeli nie chcemy by była jeszcze utworzona jego własna grupa ( o uid użytkownika ), ustawiamy USERGROUPS na ‘no’
USERGROUPS=no
ADD_EXTRA_GROUPS=1
EXTRA_GROUPS=<grupy do dodania przy tworzeniu użytkownika>
```

3. Grupy i usuwanie

- **założyć grupy użytkowników, np. grupy biuro praca.**
  `groupadd biuro`

- **dopisać użytkownika do utworzonej grupy**
  (dopisz groupe) `usermod -a -G biuro <username>`
  (nadpisz grupy) `usermod -G biuro <username>`

- **Przy pomocy deluser i delgroup proszę usunąć założone konta i grupy.**
  `userdel <username>`
  `groupdel <groupname>`

4. **Należy założyć konto systemowe. Jakie powinny być hasło i powłoka?**

```
useradd -r -sh /bin/false <username> # -r wybiera id od 100 - 999
passwd -l <username>
```

5. Modyfikacja kont

- **Przy pomocy poleceń id groups proszę sprawdzić informacje o kilku kontach użytkowników i systemowych.**
  `id <username>`
  `groups <groupname>`

- **Korzystając z polecenia chfn lub usermod zmodyfikować parametry wcześniej założonego konta: dane personalne użytkownika, nazwę konta.**
  `chfn -f "Imie Nazwisko" <username>`
  `usermod -l <new_username> <old_username>`

- **Korzystając z chsh proszę zmienić powłokę wybranego konta.**
  `chsh -s /bin/false <username>`

- **Wykorzystując polectenie ls -ln można zobaczyć numeryczne wartości właścicieli plików. Przed zmianą nazwy konta proszę oglądnąć te wartości. Operację należy powtórzyć po zmianie nazwy grupy lub konta i porównać te wartości. Czy zmiana nazwy konto lub grupy pociąga za sobą zmianę uid i gid?**
  Nie

- **Proszę zablokować wybrane konto przy pomocy passwd przez zmianę hasła**
  `passwd -l <username>`

- **Proszę zablokować wybrane konto przez zmianę powłoki:**
  `chsh -s /bin/false <username>`

6. Zakładanie partycji

- **Sprawdzić przy pomocy mount na jakim dysku (urządzeniu) znajduje się główny system plików (np. /dev/hda).**
  mount | mount -l (pierwsza linijka wskazuje gdzie zamontowany jest katalog '/')

- **Stworzyć przy pomocy dd średniej wielkości plik (np. 16MB) login.swap, gdzie login jest nazwą konta na którym sie pracuje, wypełniony zerami (plik będzie wykorzystywany do zakładania systemów plików):**
  dd if=/dev/zero of=/host/gjn.swap bs=1M count=16 # if - plik wejściowy, of - plik wyjściowy, bs - block size w bajtach, count - ile bloków ma być skopiowanych, tak więc rozmiar bliku to **bs * count** w bajtach
-

7. Zakładanie systemów plików

- **Na utworzonym w części wcześniejszej urządzeniu (np. odpowiednim loop) (lub partycji) założyć przy pomocy mkfs:**
  - **system plików ext2**
    `mkfs -t ext2 /dev/loop3`
  - **system plików minix (o ile to możliwe)**
    `mkfs -t minix /dev/loop3`
  - **system plików msdos (o ile to możliwe)**
    `mkfs -t msdos /dev/loop3`
  - **Jeżeli w punkcie 1. założono system ext2 proszę uruchomić na nim dumpe2fs i oglądnąć wyświetlane informacje. Gdzie są licznik i limit montowań?**
    `Mount count`
    `Maximum mount count`

Zajęcia 10

1. Zamykanie systemu

```
# Zamyka system
shutdown

# Zamyka system natychmiastowo
shutdown -h now

# Zamyka system po określonym czasie i wysyła komunikat
shutdown -P <czas_do_zamkniecia_w_minutach>

# Anuluje zamykanie systemu
shutdown -c
```

2. Start systemu

```
# polecenie systemowe rodziny Unix wyświetlające bufor warstwy jądra. Umożliwia m.in.
# wyświetlenie komunikatów startowych ładowania systemu.
dmesg
```

3. Init

- **Sprawdzić jaki jest domyślny tryb pracy systemu.**
  `cat /etc/inittab`

- **Sprawdzić aktualny tryb pracy systemu**
  `runlevel`

- **Co oznaczają pliki S## i K## w katalogach /etc/rc<level>.d**

  S##<skrypt> - uruchomić skrypt.
  K##<skrypt> - zatrzymać skrypt.
  Skrypty wykonywane według kolejności numerów ## i przyjmują parametry **start, stop, restart, reload** oraz **force-reload**

- Zmodyfikować strukturę plików startowych tak, by podczas przejścia na domyślny tryb pracy, jako ostatnie było uruchamiane dodtakowe wybrane polecenie (skrypt).

  Do katalogu rc##.d wystarczy dodać skrypt z najwyższym atrybutem (Snn) i uczynić go wykonywalnym, system sam zadba o wykonanie go przy starcie

  .

4. Procesy i sygnały

- **Po otwarciu dwóch oddzielnych sesji proszę w każdej z nich wywołać polecnie: ps. Jakie procesy są widoczne?**
  `ps, bash`

- **Proszę przećwiczyć użycie opcji l u m e polecenia ps.**
  `ps l - wyświetla procesy w długim formacie`
  `ps u - wyświetla procesy razem z użytkownikiem`
  `ps m - pokaż wątki po procesach`
  `ps e - pokaż zmienne środowiskowe po nazwie polecenia`

- **Jakich opcji ps należy użyć, aby obejrzeć:**
  - Proces init
    `ps x lub ps -e`

  - PID procesów
    `ps` lub `ps -e` (dla wszystkich procesów)

  - PPID procesów
    `ps -f`

  - środowisko procesów
    `ps e`

  - informacje o zużyciu pamięci przez procesy.
    `ps aux`

- **Jak przy pomocy ps i innego polecenia można oglądnąć procesy wybranego użytkownika?**
  `ps -u <user>`

- **Uruchomić w tle proces lynx, odnaleźć jego PID i usunąć proces.**
  ```
  lynx agh.edu.pl &
  ps | grep lynx
  kill -9 <pid>
  ```

- **Jak można przy pomocy jednokrotnego użycia kill usunąć wszystkie procesy w danej sesji?**
  `killall5`

- **Jaki sygnał należy wysłać do demona systemowego by przeczytał swoją konfigurację?**
  `kill -HUP <proces>`

4. Procesy i sygnały

- **Przy pomocy ps wyświetlić wartości nice procesów.**
  `ps -o pid,ni,cmd`

- **Uruchomić proces find z nice 10. Jak zmienić wartość nice tego procesu?**
  ```
  nice -n10 find
  renice -p<pid> -n<wartość>
  ```

- **Zmienić wartość nice dla powłoki w której się pracuje.**
  `renice -n<wartość> $$`

- ^_^ Uruchomić w tle 2 procesy find, z nice odpowiednio 0 i 19. Który z nich zakończy się pierwszy? Zmierzyć czas ich działania przy pomocy time.

5. At i Cron

- **Uruchomić who za godzinę**
  echo “who” | at + 1 hours

- **Uruchomić ps w niedzielę**
  echo “ps” | at sunday

- **Uruchomić df jutro o tej samej porze**
  echo “df” | at tomorrow