Automatyzacja pracy - skrypty - zaawansowane operacje administracyjne w Linux.

Cel ogólny lekcji jest związany z nauczeniem zaawansowanych operacji administracyjnych w systemie Linux, z naciskiem na automatyzację pracy za pomocą skryptów.

Cele szczegółowe lekcji:

1. Zdobycie umiejętności przekopiowania skryptu w Linuxie
2. Zdobycie umiejętności nadania atrybutu wykonywalności skryptowi
3. Zdobycie umiejętności uruchamiania skryptów
4. Nauczenie się sposobu pracy skryptów na przykładzie analizy działań skryptów 1.1 - 1.5, a także zrozumienie sposobu ich działania
5. Zrozumienie sposobu zmiany nazw plików w zadanym katalogu przez skrypt 1.1
6. Nauczenie się sposobu tworzenia plików na podstawie listy w pliku przy użyciu skryptu 1.2
7. Nauczenie się sposobu usuwania plików niebędących plikami wykonywalnymi przez skrypt 1.3
8. Nauczenie się sposobu numerowania plików na podstawie ich wielkości przy użyciu skryptu 1.4
9. Nauczenie się sposobu łączenia plików o zadanym rozszerzeniu w jeden plik przy użyciu skryptu 1.5.
10. Umiejętność podłączania do Ubuntu destop plików skryptów i przeprowadzania zaawansowanych operacji administracyjnych.
11. Poznanie i zrozumienie sposobu działania skryptów w Linuxie.
12. Umiejętność tworzenia, uruchamiania oraz modyfikacji skryptów w Linuxie.
13. Zdolność do interpretacji kodu skryptów oraz diagnozowania błędów.
14. Umiejętność samodzielnego rozwiązywania problemów przy użyciu skryptów w Linuxie.
15. Zdolność do zastosowania narzędzi automatyzacji pracy w praktycznych zastosowaniach administracyjnych.
16. Rozwój umiejętności analitycznych, logicznego myślenia i rozwiązywania problemów w kontekście systemów operacyjnych.

Wprowadzenie

Wykonaj poniższe czynności, po każdej czynności wprowadzającej nowe polecenie (nie występujące do obecnego ćwiczenia w materiałach) zapisz w zeszycie co stało się po wykonaniu polecenia, oraz sprawdź efekt wykonania polecenia.

Podłącz do Ubuntu destop jako cd plik skrypty.iso. Każdy skrypt przekopiuj a następnie nadaj atrybut wykonywalności i uruchom, skrypt lub przekopiuj skrypt z bezpośrednio z instrukcji do pliku skryptu.

Utwórz folder cw2_1718 w nim utwórz i przetestuj wszystkie skrypty.

Powtarzaj dla każdego skryptu zmieniając odpowiednio nazwę skryptu np.:

Utwórz plik skryptu `skrypt1.1`

```
chmod +x skrypt1.1
./skrypt1.1
```

Przeanalizuj pracę skryptów.

Ćwiczenie 1

Przygotowanie:

```
touch plik{,2.old}
```

Skrypt 1.1

Opis sprytu: Skrypt zmienia nazwy wszystkich plików w zadanym katalogu (parametr wywołania skryptu), do których posiadamy prawo zapisu, przez dopisanie dodatkowego członu .old. Wcześniej kasuje wszystkie pliki, które mają już takie rozszerzenie.

```
#!/bin/bash

# Przechodzenie przez wszystkie pliki w bieżącym katalogu
for nazwa in *
do
   # Sprawdzenie, czy $nazwa jest plikiem i czy jest zapisywalny
   if [ -f "$nazwa" ] && [ -w "$nazwa" ]
   then
      mv "$nazwa" "$nazwa.old"
   fi
done
```

Wywołaj polecenia:

```
./skrypt1.1
```

Skrypt 1.2

Przygotowanie:

```
echo -e "plik1.txt\nplik2.txt\nplik3.txt" > lista_plikow
mkdir katalog
```

Opis sprytu: Skrypt tworzy nowe pliki w zadanym katalogu (parametr wywołania skryptu), według listy umieszczonej w pliku (drugi parametr wywołania skryptu). Nowe pliki mają pustą zawartość. Jeżeli któryś plik już istnieje, to nie jest niszczony.

```
#!/bin/bash
if [ $# -ne 2 ]
then
      echo "Niepoprawna liczba argumentów. "
      echo "Użycie: $0 katalog plik_z_listą"
      exit 1
```

```
fi

if [ -d $1 ]
then
        if [ -f $2 ]
        then
                pliki=`cat $2`
        else
                echo "Nie ma pliku $2!"
                exit 1
        fi
else
        echo "$1 nie jest katalogiem!"
        exit 1
fi

for plik in $pliki
do
        if [ ! -e $plik ]
        then
                touch $1/$plik
        fi
done
```

Ten skrypt można wywołać w terminalu, podając dwa argumenty: ścieżkę do katalogu, w którym chcesz tworzyć pliki, oraz nazwę pliku zawierającego listę nazw plików, które mają zostać stworzone w tym katalogu. Wywołaj polecenia:

```
./skrypt1.2 ~ lista_plikow
./skrypt1.2 katalog lista_plikow
```

Sprawdź, czy zostały utworzenie pliki w ~

```
ls -la ~ | grep -E 'plik[123]'
```

Sprawdź, czy zostały utworzenie pliki w katalog

```
ls -la katalog | grep -E 'plik[123]'
```

Skrypt 1.3

Przygotowanie:

```
echo '#!/bin/bash' > skrypt1
echo 'echo "To jest skrypt 1"' >> katalog/skrypt1
echo '#!/bin/bash' > skrypt2
echo 'echo "To jest skrypt2"' >> katalog/skrypt2
chmod +x katalog/skrypt1 katalog/skrypt2
```

Skrypt usuwa wszystkie pliki w zadanym katalogu (parametr wywołania skryptu), poza plikami wykonywalnymi, mającymi ustawiony bit execute.

```
#!/bin/bash
```

```
if [ $# -ne 1 ]
then
        echo "Niepoprawna liczba argumentów. "
        echo "Użycie: $0 katalog"
        exit 1
fi

if [ -d "$1" ]; then
    biezacy_katalog=$(pwd)
    cd "$1" || exit 1  # Jeśli cd się nie powiedzie, to kończymy ze statusem 1
    for plik in *
    do
        if [ -x "$plik" ]; then
            continue
        else
            rm "$plik" -f
        fi
    done
    cd "$biezacy_katalog" || exit 1
else
    echo "$1 nie jest katalogiem!"
    exit 1
fi
```

Wywołaj polecenia:

./skrypt1.3 katalog

Sprawdź, czy w katalog pozostały pliki wykonywalne, mające ustawiony bit execute.

find katalog -type f -executable

Skrypt 1.4

Opis sprytu: Skrypt numerujący wszystkie pliki w zadanym katalogu (parametr wywołania skryptu), do których mamy prawo wykonywania (execute), przez dodanie dodatkowego członu rozszerzenia o postaci .numer_kolejny. Numeracja przebiega według wielkości plików.

```
#!/bin/bash

if [ $# -ne 1 ]
then
    echo "Niepoprawna liczba argumentów. "
    echo "Użycie: $0 katalog"
    exit 1
fi

if [ ! -d "$1" ]
then
    echo "$1 nie jest katalogiem!"
    exit 1
```

```
fi

biezacy_katalog=$(pwd)
cd "$1"
licznik=1

for plik in *
do
    if [ -d "$plik" ]
    then
        continue
    fi

    mv "$plik" "${plik}.${licznik}"
    ((licznik++))
done

cd "$biezacy_katalog"
```

Wywołaj polecenia:

./skrypt1.4 katalog

Sprawdź, czy w katalog pozostały pliki wykonywalne, mające ustawiony bit execute.

find katalog -type f -executable

Skrypt 1.5

Przygotowanie:

Utwórz dwa pliki o zadanym rozszerzeniu sh Treść każdego pliku poprzedzona jest nagłówkiem z jego nazwą.

```
#!/bin/bash

if [ ! -d "katalog" ]
then
    echo "Katalog 'katalog' nie istnieje!"
    exit 1
fi

cd "katalog"

# Utwórz pierwszy plik
nazwa_pliku1="plik1.sh"
echo "# Nagłówek pliku: $nazwa_pliku1" > "$nazwa_pliku1"
echo "Treść pliku 1..." >> "$nazwa_pliku1"

# Utwórz drugi plik
nazwa_pliku2="plik2.sh"
echo "# Nagłówek pliku: $nazwa_pliku2" > "$nazwa_pliku2"
```

echo "Treść pliku 2..." >> "$nazwa_pliku2"

echo "Utworzono pliki: $nazwa_pliku1, $nazwa_pliku2"

Opis właściwego sprytu: Skrypt łączy w jeden wszystkie pliki należące do zadanego katalogu (parametr wywołania skryptu), o zadanym rozszerzeniu (drugi parametr skryptu). Kolejność, w jakiej pliki są łączone jest nieistotna. Treść każdego pliku poprzedzona jest nagłówkiem z jego nazwą.

```
#!/bin/bash

if [ $# -ne 2 ]
then
    echo "Niepoprawna liczba argumentów. "
    echo "Użycie: $0 katalog rozszerzenie"
    exit 1
fi

if [ ! -d "$1" ]
then
    echo "$1 nie jest katalogiem!"
    exit 1
fi

biezacy_katalog=$(pwd)
cd "$1"
plik_wynikowy="polaczone.$2"

for plik in *."$2"
do
    if [ -d "$plik" ]
    then
        continue
    fi

    echo "Plik $plik:" >> "$plik_wynikowy"
    cat "$plik" >> "$plik_wynikowy"
done

cd "$biezacy_katalog"
```

Wywołaj polecenia:

./skrypt1.5 katalog sh

Sprawdź, czy w katalog powstał plik polaczone.sh i czy jego zawartością jest połączona zawartość plików znajdujących się w tym katalogu o rozszerzeniu sh.

cat -n katalog/polaczone.sh

Edytuj plik polaczone.sh w linii 7 dodaj „byłem tu”.

nano -l katalog/polaczone.sh

Skrypt 1.6

Przygotowanie:

```
echo "to jest plik plik11.txt" > plik11.txt
echo "to jest plik plik22.txt" > plik22.txt
echo "to jest plik plik33.txt" >plik33.txt
echo -e "plik11.txt\nplik22.txt\nplik33.txt" > lista_plikow2
```

Opis sprytu: Skrypt łączy w jeden pliki z listy umieszczonej w pliku o zadanej nazwie (parametr wywołania skryptu). Kolejność, w jakiej pliki są łączone - ściśle według listy. Treść każdego pliku poprzedzona jest nagłówkiem z jego nazwą. Plik wynikowy ma nazwę pliku pierwotnie zawierającego listę.

```
#!/bin/bash
if [ $# -ne 1 ]
then
        echo "Niepoprawna liczba argumentów. "
        echo "Użycie: $0 plik_z_listą"
        exit 1
fi

if [ ! -f $1 ]
then
        echo "Nie ma pliku $1!"
        exit 1
fi

ZAWARTOSC=`cat $1`
rm $1 -f
for plik in $ZAWARTOSC
do
        if [[ -f $plik ]]
        then
                echo "Plik $plik:" >> $1
                cat $plik >> $1
        fi
done
```

Wyświetlenie zawartości pliku *lista_plikow2* *(udokumentuj wycinkiem z ekranu)*

cat lista_plikow2

Wywołaj polecenia:

./skrypt1.6 lista_plikow2

Wyświetlenie zawartości pliku lista_plikow2

cat lista_plikow2

Porównaj zawartość pliku lista_plikow2 wyświetloną wcześniej *(udokumentowaną wycinkiem z ekranu)* z obecnie wyświetloną zawartością pliku lista_plikow2.

Skrypt 1.7

Przygotowanie:

W bieżącym katalogu utwórz katalog docelowy o nazwie **katalog_docelowy**

mkdir katalog_docelowy

Opis sprytu: Skrypt przesuwa wszystkie pliki z ustawionym prawem wykonywania (execute) z jednego katalogu do drugiego. Pozostałe pliki w katalogu nie są ruszane. Nazwy katalogów, źródłowego i docelowego, zadawane są jako parametry skryptu.

```
#!/bin/bash

if [ $# -ne 2 ]; then
    echo "Niepoprawna liczba argumentów."
    echo "Użycie: $0 katalog_źródłowy katalog_docelowy"
    exit 1
fi

if [[ -d $1 ]] && [[ -d $2 ]]; then
    biezacy_katalog=$(pwd)
    cd "$1"
    for plik in *; do
        if [ -d "$plik" ]; then
            continue
        else
            if [ -x "$plik" ]; then
                mv "$plik" "$2"
            fi
        fi
    done
    cd "$biezacy_katalog"
else
    echo "$1 lub $2 nie jest katalogiem!"
    exit 1
fi
```

Wywołaj polecenie:

./skrypt1.7 katalog "/home/ubuntu/cw 1718/docelowy"

Sprawdź, czy w katalogu docelowym są tylko plik z ustawionym prawem wykonywania (execute).

ls -la "/home/ubuntu/cw 1718/docelowy"

Sprawdź, czy w katalogu **katalog** nie ma plików z ustawionym prawem wykonywania (execute).

ls -la "/home/ubuntu/cw 1718/katalog"

lub

ls -la "/home/ubuntu/cw 1718/katalog" |grep x

Skrypt 1.8

Przygotowanie:

```
echo '#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Niepoprawna liczba argumentów."
    echo "Użycie: $0 katalog"
    exit 1
fi

TARGET_DIR="$1"
MAX_SUBDIRS=5
MAX_FILES=10

function create_random_files {
    local num_files=$((1 + RANDOM % MAX_FILES))
    for ((i = 1; i <= num_files; i++)); do
        touch "$1/plik$i.txt"
    done
}

function create_random_dirs {
    local num_subdirs=$((1 + RANDOM % MAX_SUBDIRS))
    for ((i = 1; i <= num_subdirs; i++)); do
        local subdir="$1/katalog$i"
        mkdir "$subdir"
        create_random_files "$subdir"
    done
}

if [ ! -d "$TARGET_DIR" ]; then
    mkdir -p "$TARGET_DIR"
fi

create_random_dirs "$TARGET_DIR"

echo "Przypadkowa struktura katalogów i plików została utworzona w: $TARGET_DIR"' > skryptp1.8
```

Nadaj atrybut wykonywalności

```
chmod +x skryptp1.8
```

Uruchom skrypt przygotowawczy

```
./skryptp1.8 "/home/ubuntu/cw 1718/katalog"
```

Opis sprytu właściwego: Skrypt pokazuje pliki z zadanego katalogu (parametr wywołania skryptu), wraz z jego podkatalogami. Zawartość podkatalogów jest listowana w postaci ścieżka dostępu względem listowanego katalogu/nazwa pliku. Liczba zagnieżdżeń podkatalogów ograniczana jest przez zmienną glebokosc.

```
#!/bin/bash

function pokaz_zawartosc_kat {
    local katalog="$1"
    local glebokosc="$2"

    for plik in "$katalog"/*; do
        if [ -f "$plik" ] || [ -d "$plik" ]; then
            echo "$(basename "$plik")"
            if [ -d "$plik" ] && [ "$glebokosc" -gt 0 ]; then
                pokaz_zawartosc_kat "$plik" "$((glebokosc - 1))"
            fi
        fi
    done
}

if [ $# -ne 2 ]; then
    echo "Niepoprawna liczba argumentów."
    echo "Użycie: $0 katalog glebokosc"
    exit 1
fi

if [ ! -d "$1" ]; then
    echo "$1 nie jest katalogiem!"
    exit 1
fi

pokaz_zawartosc_kat "$1" "$2"
```

Wywołaj polecenie:

./skrypt1.8 katalog 2

Skrypt 1.9

Opis sprytu: Skrypt usuwa wszystkie pliki puste (o zerowej wielkości) w zadanym katalogu (parametr wywołania skryptu). Skrypt tworzy w zadanym pliku listę skasowanych plików. Nie analizuje dołączeń symbolicznych.

```
#!/bin/bash

if [ $# -ne 2 ]
then
        echo "Niepoprawna liczba argumentów."
        echo "Użycie: $0 katalog plik_wynikowy"
        exit 1
fi

if [ ! -d $1 ]
then
```

```
        echo "$1 nie jest katalogiem!"
        exit 1
fi

biezacy_katalog=`pwd`
cd $1
for plik in *
do
        if [[ ! -s $plik ]] && [[ -f $plik ]] && [[ ! -L $plik ]]
        then
                echo "Usunąłem plik $plik." >> "$biezacy_katalog/$2"
                rm "$plik" -f
        fi
done

cd "$biezacy_katalog"
```

Zapisz skrypt jako skrypt1.9

Nadaj mu atrybut wykonywalności: chmod +x skrypt1.9

Uruchom go, podając katalog, w którym chcesz usunąć puste pliki, oraz nazwę pliku wynikowego, w którym będą zapisane informacje o usuniętych plikach:

./skrypt1.9 katalog" wynikowy

Upewnij się, że jesteś w tym samym katalogu, w którym znajduje się plik skrypt1.10.sh. Skrypt powinien przejść przez katalog, usuwając puste pliki i zapisując informacje o usunięciach w pliku wynikowym.

Skrypt 1.10

Skrypt porównuje zawartości dwóch zadanych katalogów (argumenty skryptu).

Przy porównaniu ignoruje podkatalogi. W wyniku wyświetla na ekranie listę plików identycznych w obu katalogach.

```
#!/bin/bash
if [ $# -ne 2 ]
then
        echo "Niepoprawna liczba argumentów. "
        echo "Użycie: $0 katalog_źródłowy katalog_docelowy"
        exit 1
fi

if [[ ! -d $1 ]]; then
        echo "$1 nie jest katalogiem!"
        exit 1
fi
if [[ ! -d $2 ]]; then
        echo "$2 nie jest katalogiem!"
        exit 1
fi
```

```
WYKAZ1=`ls $1`
WYKAZ2=`ls $2`

for plik1 in $WYKAZ1
do
	for plik2 in $WYKAZ2
	do
		if [[ ! -d $1/$plik1 ]] && [[ ! -d $2/$plik2 ]]
		then
			diff $1/$plik1 $2/$plik2 > /dev/null
			if [[ $? -eq 0 ]] && [[ $plik1 = $plik2 ]]
			then
				echo "Plik $plik1 jest taki sam w obu katalogach."
			fi
		fi
	done
done
```

Zapisz skrypt jako skrypt1.10

Nadaj mu atrybut wykonywalności: chmod +x skrypt1.10

Wywołaj polecenie:

./skrypt1.10 katalog docelowy

Skrypt ./skrypt1.10 działa poprawnie w tym przypadku, ponieważ nie wyświetla żadnych komunikatów błędu ani informacji o tym, że znalazł pliki o takiej samej zawartości w obu katalogach. Jeśli to jest zamierzony rezultat, to wszystko jest w porządku. Skrypt porównuje zawartość plików w dwóch katalogach i informuje, jeśli znajdzie plik o takiej samej zawartości w obu katalogach.

Skrypt 1.11

Przygotowanie:

```
echo "to jest plik plik111.txt" > katalog/plik111.txt
echo "to jest plik plik222.txt" > katalog/plik222.txt
echo "to jest plik plik333.txt" >plik333.txt
echo -e "plik111.txt\nplik222.txt\nplik333.txt" > plik_z_lista
```

Skrypt porównuje zawartości zadanego katalogu z listą plików (nazwa katalogu i pliku z listą zadawana jest w argumentach skryptu). Skrypt generuje listę plików brakujących w katalogu i takich, które nie są na liście.

```
#!/bin/bash
if [ $# -ne 2 ]
then
	echo "Niepoprawna liczba argumentów. "
	echo "Użycie: $0 katalog_źródłowy plik_z_listą"
	exit 1
```

```
fi

if [[ ! -d $1 ]]; then
        echo "$1 nie jest katalogiem!"
        exit 1
fi

if [[ ! -f $2 ]]; then
        echo "Nie ma pliku $2!"
        exit 1
fi

WYKAZ1=`ls $1`
WYKAZ2=`cat $2`

echo "Pliki z listy których nie ma w katalogu:"
for plik1 in $WYKAZ2
do
        ZNALEZIONO=0
        for plik2 in $WYKAZ1
        do
                if [[ $plik1 = $plik2 ]]
                        then
                        ZNALEZIONO=1
                fi
        done
        if [[ ZNALEZIONO -eq 0 ]]
        then
                echo $plik1
        fi
done

echo "Pliki których nie ma na liście, a są w katalogu:"
for plik1 in $WYKAZ1
do
        ZNALEZIONO=0
        for plik2 in $WYKAZ2
        do
                if [[ $plik1 = $plik2 ]]
                        then
                        ZNALEZIONO=1
                fi
        done
        if [[ ZNALEZIONO -eq 0 ]]
        then
                echo $plik1
        fi
done
```

Wykonaj następujące kroki:

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt1.11

Uruchom skrypt, podając jako argumenty ścieżkę do katalogu źródłowego oraz ścieżkę do pliku z listą plików:

./skrypt1.11 katalog plik_z_lista

Gdzie:

skrypt1.11 to nazwa pliku skryptu,

katalog to ścieżka do katalogu źródłowego,

plik_z_lista to ścieżka do pliku zawierającego listę plików.

Upewnij się, że ścieżki są poprawne i że plik skryptu znajduje się w tym samym katalogu, w którym się znajdujesz, lub podaj pełną ścieżkę do tego pliku.

Skrypt 1.12

Skrypt usuwana wszystkie podkatalogi zadanego katalogu (parametr wywołania skryptu). Zawartość tych podkatalogów przenoszona jest do katalogu nadrzędnego. Usuwanie dotyczy tylko jednego poziomu podkatalogów.

```
#!/bin/bash
if [ $# -ne 1 ]
then
        echo "Niepoprawna liczba argumentów. "
        echo "Użycie: $0 katalog"
        exit 1
fi

if [[ ! -d $1 ]]; then
        echo "$1 nie jest katalogiem!"
        exit 1
fi

WYKAZ=`ls $1`

for plik in $WYKAZ
do
        if [[ -d $plik ]]
        then
                mv $1/$plik/* $1
                rmdir $1/$plik
        fi
done
```

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt1.12

Wywołanie tego skryptu będzie wyglądać następująco:

./skrypt1.12 katalog

Gdzie skrypt1.12 to nazwa twojego skryptu, a katalog to ścieżka do katalogu, w którym chcesz przenieść zawartość podkatalogów do głównego katalogu.

Skrypt 1.13

Przygotowanie:

echo "Treść twojej wiadomości" > wiadomosc.txt

Skrypt wysyła pocztę elektroniczną do wszystkich użytkowników systemu.
Treść listu zadana zawartością pliku, będącego parametrem wywołania skryptu.
Listę lokalnych użytkowników uzyskiwana jest za pomocą listowania zawartość katalogu systemowego /var/spool/mail (najprostsza metoda, ale nie najlepsza).

```
#!/bin/bash
if [ $# -ne 1 ]
then
        echo "Niepoprawna liczba argumentów. "
        echo "Użycie: $0 plik_z_wiadomością"
        exit 1
fi

if [[ ! -e $1 ]]; then
        echo "Nie ma pliku $1!"
        exit 1
fi

WYKAZ=`ls /var/spool/mail`

for adresat in $WYKAZ
do
        cat $1 | mail -s "Temat wiadomości" $adresat
done
```

Ten skrypt będzie działać tylko wtedy, gdy masz odpowiednie uprawnienia do wysyłania wiadomości e-mail, a treść wiadomości będzie niezawodna i pozbawiona potencjalnych zagrożeń.

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt1.13

Aby wywołać ten skrypt, musisz podać jako argument nazwę pliku z treścią wiadomości, którą chcesz wysłać. Treść wiadomości znajduje się w pliku o nazwie wiadomosc.txt. Wywołaj skrypt:

./skrypt1.13 wiadomosc.txt

Upewnij się, że znajdujesz się w katalogu, w którym znajduje się skrypt, lub podaj pełną ścieżkę do skryptu, jeśli jesteś w innym katalogu.

Upewnij się, że plik wiadomosc.txt znajduje się w tym samym katalogu co skrypt, który wcześniej utworzyliśmy. Następnie możesz użyć wcześniej podanego polecenia, aby wywołać skrypt i wysłać wiadomość.

W skrypcie skrypt1.13 użyłeś polecenia cat $1 | mail, które przesyła zawartość pliku do programu pocztowego mail, jednakże nie jesteś w stanie w ten sposób śledzić, czy wiadomość została dostarczona czy też nie. Polecenie mail standardowo wysyła wiadomość do lokalnej skrzynki pocztowej użytkownika na systemie.

Aby sprawdzić, czy wiadomość została dostarczona, możesz zalogować się na konto, na które próbujesz wysłać wiadomość, i sprawdzić zawartość skrzynki pocztowej. Możesz to zrobić na przykład za pomocą programu mail:

```
mail
```

To polecenie pokaże ci listę nieprzeczytanych wiadomości w skrzynce pocztowej. Jeśli zobaczysz swoją wiadomość, to oznacza, że została ona dostarczona.

Jeśli masz dostęp do innej skrzynki pocztowej (np. Gmail) i chcesz sprawdzić, czy wiadomość została dostarczona tam, będziesz musiał zalogować się na swoje konto pocztowe przez przeglądarkę internetową lub inne narzędzia do zarządzania pocztą.

```
root@ubuntu-VirtualBox:/home/ubuntu/cw 1718# ./skrypt1.13 wiadomosc.txt
root@ubuntu-VirtualBox:/home/ubuntu/cw 1718# mail
"/var/mail/root": 1 wiadomość   1 nowa
>N   1 root               śro sie 16 19:5  14/463   Temat wiadomości
?
```

Aby uyzyskać powyższy wycinek należy zainstalować mail (apt -y install mailutils) a następnie ponownie uruchomic skrypt ./skrypt1.13 wiadomosc.txt i oczywiście uruchomić mail a potem Ctrl + Z aby wycofać się.

Skrypt 1.14

Skrypt ustawia na aktualny czas ostatniej modyfikacji wszystkich plików zadanego katalogu (parametr wywołania skryptu), do których posiadamy prawo do zapisu/modyfikacji.

```
#!/bin/bash
if [ $# -ne 1 ]
then
        echo "Niepoprawna liczba argumentów. "
        echo "Użycie: $0 katalog"
        exit 1
fi
```

```
if [[ ! -d $1 ]]; then
        echo "Nie ma katalogu $1!"
        exit 1
fi

biezacy_katalog=`pwd`
cd $1
for plik in *
do
        if [[ ! -d $plik ]]
        then
                touch -m $plik
        fi
done
cd "$biezacy_katalog"
```

Skrypt ma na celu aktualizowanie czasu modyfikacji plików w określonym katalogu. Skrypt najpierw sprawdza liczbę argumentów, a następnie czy podany argument jest istniejącym katalogiem. Następnie przechodzi do tego katalogu, iteruje przez pliki wewnątrz niego i używa polecenia touch -m do zaktualizowania czasu modyfikacji plików, które nie są katalogami.

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

```
chmod +x skrypt1.14
```

Wywołaj go, podając jako argument katalog, w którym chcesz zaktualizować czas modyfikacji plików:

```
./skrypt1.14 katalog
```

Zamień "katalog" na właściwą ścieżkę do katalogu, który chcesz zaktualizować. Skrypt przejdzie przez wszystkie pliki wewnątrz tego katalogu i zaktualizuje ich czas modyfikacji.

To polecenie wyświetli listę plików wraz z informacją o czasie ostatniego dostępu (odczytu) do plików.

```
ls -l --time=atime katalog
```

Skrypt 1.15

Skrypt zlicza wszystkie pliki w zadanym katalogu (parametr wywołania skryptu), do których ustawione jest prawo do wykonania (*execute*).

```
#!/bin/bash
if [ $# -ne 1 ]
then
        echo "Niepoprawna liczba argumentów. "
        echo "Użycie: $0 katalog"
        exit 1
fi

if [[ ! -d $1 ]]; then
        echo "Nie ma katalogu $1!"
```

```
        exit 1
fi

biezacy_katalog=`pwd`
cd $1
licznik=0
for plik in *
do
        if [[ -x $plik ]]&&[[ -f $plik ]]
        then
                let licznik=licznik+1
        fi
done
cd $biezacy_katalog
echo "Liczba plików z prawem do wykonywania w podanym katalogu: $licznik."
```

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt1.15

Aby wywołać ten skrypt, otwórz terminal i wpisz:

./skrypt1.15 katalog

Gdzie skrypt1.15 to nazwa pliku skryptu, a katalog to ścieżka do katalogu, który chcesz zbadać pod kątem plików z prawem do wykonywania.

Ćwiczenia 2

Uruchom oraz przeanalizuj poniższe skrypty.

1) Program wczytujący, wyświetlający i zapisujący do pliku zmienną

```
#!/bin/bash
# Program wczytujący, wyświetlający i zapisujący do pliku zmienną

echo "Jak masz na imię?"
read zmienna
echo "Twoje imię to: $zmienna"
touch wizytowka.txt
echo "$zmienna" >> wizytowka.txt
```

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt2.1

Wywołaj w terminalu, przechodząc do katalogu, w którym się znajduje, i wpisując:

./skrypt2.1

Skrypt poprosi cię o imię, a następnie zapisze je do pliku "wizytowka.txt".

cat wizytowka.txt

2) program z instrukcja if

```
#!/bin/bash
#program z instrukcja if
haslo="test";
echo "Podaj haslo:"
read czytaj
if [ $haslo = $czytaj ]; then
echo "Hasło przyjete";
else
echo "$czytaj nie jest prawidłowym hasłem"
fi
```

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt2.2

Wywołaj w terminalu, przechodząc do katalogu, w którym się znajduje, i wpisując:

./skrypt2.2

Skrypt poprosi cię o podanie hasła, a następnie wyświetli komunikat "Hasło przyjęte", jeśli hasło jest zgodne z tym, które ustawiłeś jako haslo. W przeciwnym razie wyświetli komunikat "Podane hasło nie jest prawidłowym hasłem".

3) program tworzący katalogi i pliki

```
#!/bin/bash
#program tworzacy katalogi i pliki
mkdir tydzien;
mkdir tydzien/poniedzialek;
mkdir tydzien/wtorek;
touch tydzien/wtorek/plik1.txt;
touch tydzien/wtorek/plik2.txt;
touch tydzien/wtorek/plik3.txt;
echo "Zawartosc katalogu tydzien:"
ls tydzien
echo "Zawartosc katalogu wtorek:"
ls -a tydzien/wtorek
chmod 000 tydzien/wtorek/plik2.txt
```

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt2.3

Wywołaj w terminalu, przechodząc do katalogu, w którym się znajduje, i wpisując:

./skrypt2.3

Skrypt utworzy strukturę katalogów i plików, wyświetli ich zawartość, a następnie zmieni prawa dostępu dla pliku plik2.txt w katalogu wtorek.

4) procesy uruchomione w systemie

```
#!/bin/bash
echo "procesy uruchomione w systemie:"
ps -A | more
echo "$USER ktory proces zabic? - podaj PID:"
read nr;
kill $nr
echo "Zabiłem proces $nr "
```

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt2.4

Wywołaj w terminalu, przechodząc do katalogu, w którym się znajduje, i wpisując:

./skrypt2.4

Skrypt wykonuje następujące kroki:

1) Wyświetla listę wszystkich procesów uruchomionych w systemie za pomocą polecenia ps -A | more.

2) Prosi użytkownika o podanie numeru PID (identyfikatora procesu) procesu, który ma zostać zabicie.

3) Wykonuje komendę kill $nr aby zabić wybrany proces, gdzie $nr to numer PID podany przez użytkownika.

4) Wyświetla komunikat informujący, że został zabicie proces o numerze PID podanym przez użytkownika.

Przykładowy scenariusz działania skryptu:

procesy uruchomione w systemie:

```
  PID TTY          TIME CMD
    1 ?        00:00:02 systemd
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
   ...
```

$USER ktory proces zabic? - podaj PID:

12345

Zabiłem proces 12345

Pamiętaj jednak, że zabijanie procesów może mieć wpływ na działanie systemu i aplikacji, więc upewnij się, że wiesz, co robisz.

5) program z instrukcja wyboru case

```
#!/bin/bash
echo "Podaj cyfrę dnia tygodnia"
read d
case "$d" in
   "1") echo "Poniedzialek" ;;
   "2") echo "Wtorek" ;;
   "3") echo "Środa" ;;
   "4") echo "Czwartek" ;;
   "5") echo "Piątek" ;;
   "6") echo "Sobota" ;;
   "7") echo "Niedziela" ;;
   *) echo "Zły wybór"
esac
```

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt2.5

Wywołaj w terminalu, przechodząc do katalogu, w którym się znajduje, i wpisując:

./skrypt2.5

Jak działa skrypt:

1) Wyświetla komunikat "Podaj cyfrę dnia tygodnia".

2) Czyta wartość cyfry wprowadzonej przez użytkownika do zmiennej d.

3) Wykonuje instrukcję case:

   a. Jeśli wartość d jest równa "1", wyświetla "Poniedziałek".

   b. Jeśli wartość d jest równa "2", wyświetla "Wtorek".

   c. ...

   d. Jeśli wartość d jest równa "7", wyświetla "Niedziela".

   e. W przypadku innych wartości d wyświetla "Zły wybór".

Przykładowy scenariusz działania skryptu:

Podaj cyfrę dnia tygodnia

3

Środa

Pamiętaj, że przy wykorzystaniu instrukcji case, jak w tym przypadku, warto dbać o poprawność przyporządkowań i uwzględniać wszystkie możliwe wartości wejściowe.

4) Sumowanie liczb

```
#!/bin/bash
echo "podaj 1 liczbe: "
read liczba1
echo "podaj 2 liczbe:"
read liczba2
let suma=$liczba1+$liczba2;
echo "Suma liczb = $suma"
```

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt2.6

Wywołaj w terminalu, przechodząc do katalogu, w którym się znajduje, i wpisując:

./skrypt2.6

Jak działa skrypt:

1. Skrypt zaczyna się od deklaracji interpretera, którym jest Bash: #!/bin/bash.
2. Następnie skrypt wyświetla tekst "podaj 1 liczbe:" i oczekuje na wprowadzenie pierwszej liczby przez użytkownika przy pomocy polecenia read liczba1.
3. Następnie skrypt wyświetla tekst "podaj 2 liczbe:" i oczekuje na wprowadzenie drugiej liczby przez użytkownika przy pomocy polecenia read liczba2.
4. Skrypt oblicza sumę dwóch wprowadzonych liczb i przypisuje wynik do zmiennej suma za pomocą polecenia let suma=$liczba1+$liczba2.
5. Na koniec skrypt wyświetla wynik sumy, korzystając z polecenia echo "Suma liczb = $suma".

Podsumowując, ten skrypt prosi użytkownika o wprowadzenie dwóch liczb, następnie je sumuje i wyświetla wynik na ekranie.

Ćwiczenia 3

Wyszukiwanie i liczenie liczby plików i katalogów z prawami r, w, x dla zalogowanego użytkownika.
Problem: W jaki sposób obliczyć ilość plików i katalogów rekurencyjnie z podkatalogami danego katalogu i wyświetlić ilość plików i katalogów z danymi prawami r, w, x dla zalogowanego użytkownika?
Rozwiązaniem jest skrypt poniżej, który należy uruchomić i przeanalizować.

```
#!/bin/sh

dir=/
temp=`mktemp`
ilekat=0
```

```
ileplik=0
ile=0
xplik=0
rplik=0
wplik=0
xkat=0
rkat=0
wkat=0

if test "$1" = "-help"
then
echo prawa [-f][-d][katalog]
echo skrypt oblicza ilosc plikow i katalogow rekurencyjnie z podkatalogami od podanego katalogu
echo i wyswietla ilosc plikow i katalogow z danymi prawami r,w,x dla zalogowanego uzytkownika
echo -f wyswietlaj statystyke dla plikow
echo -d wyswietlaj statystyke dla katalogow
elif test $# -eq 0
then
echo brak parametrow
elif test $# -gt 3
then
echo za duzo parametrow
elif test "$1" != "-d" && test "$1" != "-f"
then
echo nieznany parametr $1
elif test $# -eq 2 && test ! -d $2 && test "$2" != "-f" && test "$2" != "-d"
then
echo bledny parametr
elif test "$2" != "-d" && test "$2" != "-f" && test "$2" != "" && test ! -d $2
then
echo nieznany parametr $2
elif test $# -eq 3 && test -d $2 && test -d $3
then
echo bledne parametry
elif test $# -eq 3 && test !  -d $3
then
echo $3 nie jest katalogiem

else
if test -d $3
then
dir=$3
fi
if test -d $2
then
dir=$2
fi

find $dir -name '*' -print > $temp
```

```
while read a
do

ile=`expr $ile + 1`
if test  -d $a
then

ilekat=`expr $ilekat + 1`
if test -x $a
then
xkat=`expr $xkat + 1`
fi
if test -r $a
then
rkat=`expr $rkat + 1`
fi
if test -w $a
then
wkat=`expr $wkat + 1`
fi

else
ileplik=`expr $ileplik + 1`

if test -x $a
then
xplik=`expr $xplik + 1`
fi
if test -r $a
then
rplik=`expr $rplik + 1`
fi
if test -w $a
then
wplik=`expr $wplik + 1`
fi

fi

done < $temp

if test "$1" = "-f" || test "$2" = "-f"
then
echo przeanalizowanych plikow: $ileplik
echo ilosc plikow wykonywalnych: $xplik
echo ilosc plikow do odczytu: $rplik
echo ilosc plikow do pisania: $wplik
fi
if test "$1" = "-d" || test "$2" = "-d"
then
```

```
echo przeanalizowanych katalogow: $ilekat
echo ilosc katalogow z prawem wykonywania: $xkat
echo ilosc katalogow z prawem do odczytu: $rkat
echo ilosc katalogow z prawem do zapisu: $wkat
fi
fi
```

Ten skrypt ma za zadanie analizować strukturę katalogów i plików, liczyć ilość plików i katalogów, a także wyświetlać statystyki dotyczące uprawnień do wykonywania, odczytu i zapisu.

Kilka kluczowych punktów w działaniu tego skryptu:

1. Skrypt sprawdza podane argumenty w celu określenia, czy ma wyświetlić statystyki dla plików (-f), katalogów (-d) lub oboje.
2. Skrypt korzysta z pętli while read a w celu przetwarzania każdego znalezionego elementu (pliku lub katalogu).
3. Dla każdego elementu (pliku lub katalogu), skrypt sprawdza, czy jest to katalog lub plik, oraz czy ma prawa do wykonywania, odczytu i zapisu. Na podstawie tych informacji zwiększa odpowiednie liczniki.
4. Na podstawie opcji podanych przy wywołaniu skryptu (-f lub -d) oraz zebranych statystyk, skrypt wyświetla informacje dotyczące ilości plików/katalogów i praw do nich.

Skrypt jest dosyć obszerny i rozbudowany, aby analizować różne warunki i opcje. Może być używany do analizy struktury plików i katalogów oraz monitorowania uprawnień dostępu.

Uwagi: Skrypt oblicza ilość plików i katalogów rekurencyjnie z podkatalogami podanego katalogu
i wyświetla ilość plików i katalogów z danymi prawami r, w, x dla zalogowanego użytkownika
Opcje:
-f wyświetlaj statystykę dla plików
-d wyświetlaj statystykę dla katalogów
-help wyświetla pomoc skryptu

Upewnij się, że plik skryptu ma prawa wykonywalności. Jeśli nie ma, możesz nadać je za pomocą polecenia:

chmod +x skrypt3

Przykładowe wywołania skryptu o nazwie skrypt3:

1. Wyświetlenie statystyk dla plików:

./skrypt3 -f

2. Wyświetlenie statystyk dla katalogów:

./skrypt3 -d

3. Wyświetlenie statystyk dla plików i katalogów:

./skrypt3 -f -d

4. Wyświetlenie statystyk dla określonego katalogu:

./skrypt3 -d /sciezka/do/katalogu

5. Wyświetlenie statystyk dla określonego pliku:

./skrypt3 -f /sciezka/do/pliku

6. Wyświetlenie pomocy:

./skrypt3 -help

Ćwiczenia 4

Napisz skrypt dla powłoki Bash, który wykona następujące czynności:

1. W aktualnym folderze utworzy foldery o nazwach „**backup**" i „**Twoje imię**".
2. W folderze „**Twoje imię**" utworzy 3 pliki o dowolnej zawartości (pliki nie mogą być puste).
3. Utworzy w folderze „**backup**" kopię bezpieczeństwa folderu „Twoje imię" wraz z zawartymi w nim plikami pod nazwą „Twoje imię - kopia.tgz". Kopię należy utworzyć programem tar.
4. Założy konta użytkowników **user1** do **user5** z hasłem zaq1@WSX
5. Założy grupę użytkowników **sekretariat** za pomocą jednej komendy.
6. Doda użytkowników **user3** i **user5** do grupy **sekretariat**.
7. Zapyta o nazwy użytkowników, których należy dodać do grupy **sekretariat,** jeżeli podany zostanie dotychczasowy członek grupy **sekretariat** to odmówi dodania użytkownika do grupy, a jeżeli podany użytkownik nie będzie należał do tej pory do grupy to doda wskazanych użytkowników w tym przypadku podaj **user2** i **user4** do grupy **sekretariat**.

Podsumowanie:

Po wykonaniu wszystkich czynności z powyższej instrukcji przeczytaj ponownie z zrozumieniem cel ogólny i cele szczegółowe, które znajdują się na pierwszej stronie instrukcji. Jeżeli one zostały niezrealizowane to powtarzaj wykonie tej instrukcji w szkole lub/i w domu do momentu zrealizowania.