	Państwowa Wyższa Szkoła Zawodowa w Nysie			Wydział Nauk Technicznych	
	Laboratorium Podstaw Systemów Komputerowych				
Kierunek:	Informatyka	Rok studiów nr:	1	Semestr nr:	2
Rok akademicki:	2020/2021	Grupa administracyjna:	L5_	Grupa ćwiczeniowa:	L5g1_

SPRAWOZDANIE

Nr ćwiczenia	Temat ćwiczenia
Nr wg listy	Strumienie standardowe, filtry i potoki – część I
Termin złożenia sprawozdania	
Termin wg listy	
Data faktycznego złożenia sprawozdania	
(nie wypełniaj)	

Wykonawcy	Nazwisko	Imię	Nr indeksu	Ocena
	Roszak	Damian		(Nie wypełniane w trybie online)
				(Nie wypełniane w trybie online)

Uwaga: Umieszczenie danych osobowych wykonawców stanowi grupowe i nieodwołalne oświadczenie, że są oni/one (i tylko oni/one) współautorami przedstawionego sprawozdania. Późniejsza zmiana składu zespołu wykonawców nie będzie możliwa.

Nie wypełniać przy składaniu online

Data i podpis prowadzącego ćwiczenia

Wymagania typograficzne

- Tekst główny (w ramkach) należy składać czcionką normalną typu Times 12 pkt.
- Zawartość plików, nazwy ścieżek w systemie plików, polecenia wydawane z konsoli i uzyskiwane odpowiedzi systemu/aplikacji oraz kopie tabulogramów interakcji z powłoką należy składać czcionką normalną typu **`Courier 11 pkt`**. Należy zachować wygląd, w tym pozycjonowanie tekstu.
- Nazwy pozycji menu w programach i nazwy przycisków ekranowych należy składać czcionką pogrubioną typu **Arial 11 pkt**.
- Wykluczone jest zamieszczanie ilustracji graficznych z ciemnym tłem. Tekst powinien z tłem wyraźnie kontrastować.

1. Temat ćwiczenia

(kopia tematu instrukcji, identyczna jak tytuł sprawozdania)

2. Zakres ćwiczenia

Streszczenie treści ćwiczenia oraz ustalenia prowadzącego zajęcia dotyczące wyboru funkcji badanego programu, zastosowanego algorytmu, zbioru przetwarzanych danych, precyzji przedstawienia liczb, liczby wątków i cykli obliczeń, sposobu prezentacji wyników, itp.)

3. Środowisko realizacji ćwiczenia

(architektura logiczna systemu – sprzęt, elementy składowe, ich cechy i sposób wzajemnego połączenia, schematy; wykorzystywane języki, oprogramowanie, biblioteki, skrypty powłokowe, zasoby sieciowe i dokumentacja)

4. Przebieg ćwiczenia i uzyskane wyniki

(przedstawienie czynności wykonanych w ramach realizacji ćwiczenia, w kolejności określonej treścią instrukcji. Dla każdego punktu instrukcji należy przedstawić: nr i tytuł tego punktu, cel działania, sposób wykonania, otrzymany rezultat i jego ocenę). Wymagana jest 100% chronologia zadań, czynności i uzyskanych rezultatów.

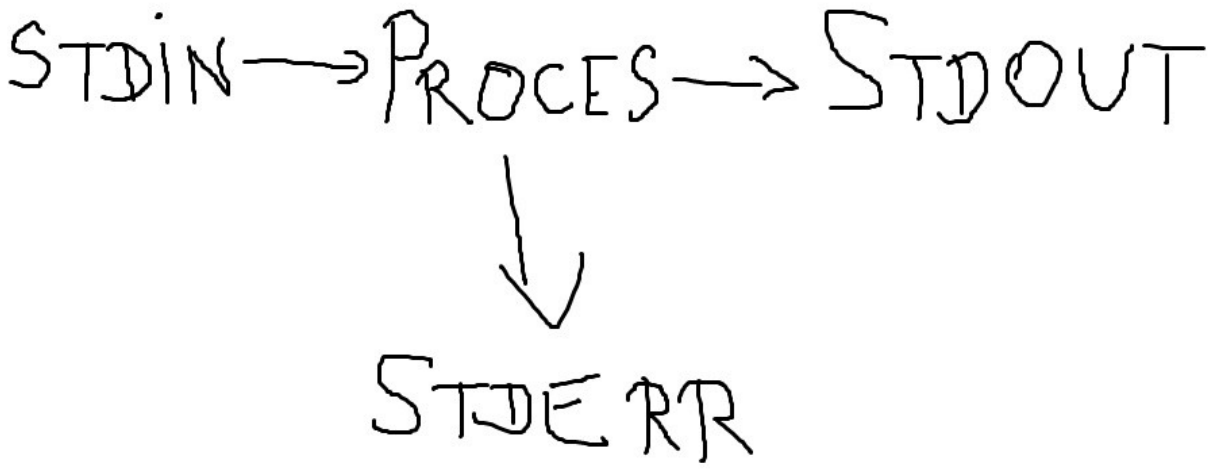
4.1 Zadanie nr 1

- 4.1.1 Nr i treść polecenia wg instrukcji
- 4.1.2 Cel czynności
- 4.1.3 Sposób i rezultat wykonania polecenia (np. polecenia wydane na konsoli i odpowiedź systemu/aplikacji, w postaci wycinka zarejestrowanego logu konwersacji terminalowej w formacie tekstowym). Dopuszcza się zamieszczenie fragmentu zrzutu ekranowego. W każdym przypadku obraz rezultatu ma obejmować wykonania wyłącznie danego punktu (a nie wszystko, co widać w oknie terminala lub konsoli). Log konwersacji musi zawierać następujące bezpośrednio po niej zaproszenie (tzw. *prompt*) powłoki.
- 4.1.4 Ocena/wnioski/komentarze dotyczące wykonania danego zadania.

4.2 Zadanie nr 2

- 4.2.1 (analogicznie jak wyżej)
- 4.2.2 ...
- 4.2.3 ...
- 4.2.4

Każdy uruchomiony w Linuksie proces pobiera skądś dane, gdzieś wysyła wyniki swojego działania i komunikaty o błędach. Dane przesyłane są między urządzeniami w postaci strumieni. Strumienie danych przypisane każdemu procesowi są pokazane na rysunku:



`stdin`(standard input), czyli standardowe wejście, z którego proces pobiera dane (domyślnie jest to klawiatura),

`stdout`(standard otuput) to standardowe wyjście, z którego wysyłany jest wynik działania procesu (domyślnie jest to ekran),

`stderr`(standard error) to standardowe wyjście błędów, gdzie trafiają wszystkie komunikaty o błędach (domyślnie ekran).

Linux wszystkie urządzenia traktuje jak pliki, niezależnie od tego, czy to jest plik, folder, klawiatura, ekran itp., czy strumień. Powłoka Linuksa identyfikuje je za pomocą przyporządkowanych im liczb całkowitych, tak zwanych deskryptorów plików:

0 to plik, z którego proces pobiera dane (`stdin`),
1 to plik, do którego proces zapisuje wyniki swojego działania (`stdout`),
2 to plik, do którego trafiają komunikaty o błędach (`stderr`),

Za pomocą operatorów przypisania można manipulować strumieniami, poprzez przypisanie deskryptorów: 0,1,2 innym plikom, niż tym, które reprezentują klawiaturę i ekran.

Standardowe wejście i wyjście (strumienie danych) możemy przekierować. Do tego celu przygotowano trzy operatory:

**znak „<" ** umożliwia przekierowanie zawartości pliku do standardowego wyjścia, np. `more < plik`,

**znak „>" ** umożliwia przekierowanie strumienia danych ze standardowego wyjścia do pliku; jeżeli plik istnieje, to jego poprzednia zawartość zostaje usunięta, np. `ls > plik`,

**znaki „>>" ** umożliwiają przekierowanie strumienia danych ze standardowego wyjścia do pliku; jeżeli plik istnieje, to nowe dane zostają dopisane na końcu pliku.
Jako standardowe wejście można zamiast klawiatury użyć pliku`: < plik`

Wynik jakiegoś polecenia można wysłać do pliku, a nie na ekran. Do tego celu używa się operatora: `> plik`

Przykład: `ls -la /usr/bin > ~/wynik`

Rezultat działania polecenia `ls -la /usr/bin` trafi do pliku o nazwie `wynik` jeśli wcześniej nie istniał plik o takiej samej nazwie, to zostanie utworzony, jeśli istniał, cała jego poprzednia zawartość zostanie nadpisana.

Jeśli chcemy, aby dane wyjściowe dopisywane były na końcu pliku, bez wymazywania jego wcześniejszej zawartości, stosujemy operator:

```
>> plik
```

Do pliku można przekierować także strumień diagnostyczny:

```
polecenie> plik
```

Za pomocą polecenia `touch` tworzymy 3 pliki tekstowe o nazwach kolejno: a.txt, b.txt, c.txt.

```
[root@centos75 ~]# touch bspl({a,b,c}.txt
```

Następnie wypełniamy pliki teksem za pomocą operatora strumienia stdout oraz funkcji `echo`:

```
[root@centos75 ~]# echo "Lorem
> ipsum
> " > c.txt
```

Używając polecenia `cat –help` dowiadujemy się jakie jest działanie polecenia `cat` dla różnych parametrów. Służy ono przede wszystkim łączeniu i przekierowywaniu plików na odpowiednie strumienie.

Łączymy pliki wyświetlając je na terminalu za pomocą polecenia:

```
cat {a,b,c}.txt
```

Powtarzamy podpunkt a) przekierowując strumień na plik d.txt:

```
cat {a,b,c}.txt > d.txt
```

Sprawdzamy poprawność wykonania polecenia za pomocą `cat d.txt` co zwraca nam zawartość pliku z treścią w odpowiedniej kolejności.

Tworzymy nowy plik o nazwie e.txt i przepisujemy do niego wprowadzenie do zadania:

```
[root@centos75 ~]# echo  "Przedmiotem cwiczenia jest zapoznanie się
z mechanizmem standardowych strumieni wejscia/wyjscia i sposobami
jego uzycia. " > e.txt
```

Za pomocą polecenia `cat` wyświetlamy zawartość pliku e.txt:

```
[root@centos75 ~]# cat e.txt
Przedmiotem cwiczenia jest zapoznanie się z mechanizmem standardo-
wych strumieni wejscia/wyjscia i sposobami jego uzycia.
[root@centos75 ~]#
```

Przy użyciu polecenia cat kopiujemy zawartość pliku e.txt do nowego pliku f.txt:

```
[root@centos75 ~]# cat e.txt > f.txt
```

Tworzymy plik auto.sh, którego celem jest automatyczne tworzenie listy 5 utworów muzycznych:

```
[root@centos75 ~]# echo "echo \"Britney Spears – Oops I did again
> Christina Aguilera – Geany in the bottle
> Vivaldi - Wiosna
> Kult - Prosto
> Kult – Układ zamknięty
> \" >muzyka.txt
> " > auto.sh
[root@centos75 ~]#
```

Następnie uruchamiamy plik przy użyciu polecenia:

```
sh auto.sh
```

lub

```
bash auto.sh
```

5. Wnioski z przeprowadzonych prac

(podsumowanie celu ćwiczenia i osiągniętych wyników, wnioski dotyczące zastosowanych środków programowych i uzyskanych wyników, samoocena stopnia osiągnięcia celu ćwiczenia)

6. Inne uwagi