**ŽILINSKÁ UNIVERZITA**
Fakulta riadenia a informatiky


# OPERAČNÉ SYSTÉMY

## PIPELINE SYSTEM

Erik Poliaček

# Zadanie:

Implementujte aplikáciu, ktorá bude spracovávať svoj štandardný vstup pomocou sekvencie "textových" operácií a výsledok vypíše na štandardný výstup, s nasledujúcimi vlastnosťami:
Jednotlivé operácie budú vykonávane v samostatných podprocesoch alebo vláknach, ktoré budú súčasťou jednej aplikácie. Proces (vlákno) P0 bude zodpovedný za načítanie štandardného vstupu a odovzdanie dát procesu P1. Proces (vlákno) Pn+1 bude zodpovedný za prevzatie dát od procesu Pn a ich vypísanie na obrazovku:

$P_0$ -> $P_1$ -> $P_2$ -> ...-> $P_{n-1}$

Procesy (vlákna) P1 až Pn budú pracovať na základe nasledujúceho algoritmu:

```
while(true) {
   nacitaj_data_od_predchadzajuceho();
   vykonaj_operaciu_na_datach();
   posli_data_nasledujucemu();
}
```

Nesmie existovať žiadne explicitné horné ohraničenie objemu dát, ktoré je schopná aplikácia spracovať. Procesy musia bežať (pseudo) paralelne, t.j. jednotlivé operácie musia pracovať s čo najmenšími jednotkami dát (ktoré majú pre danú operáciu zmysel: znak, slovo, atď. ) a hneď po spracovaní ich odoslať nasledujúcemu procesu a načítať ďalšie dáta od predchodcu. Prostriedok na prenos údajov medzi procesmi (pipe, zdielanú pamäť, atď.) si môžete zvoliť sami.

Operácie, ktoré je potrebné implementovať:

- UPPERCASE - Všetky znaky (pri ktorých to má význam) na vstupe sú prevedené na veľké,
- LOWERCASE - Všetky znaky (pri ktorých to má význam) na vstupe sú prevedené na malé,
- CAPITALIZE - Prvý znak v slove je po transformácii veľký, ostatné znaky sú malé, t.j. "slovo" -> "Slovo", "sLOvo" -> "Slovo",
- REVWORD - výpis písmena v slove v obrátenom poradí. t.j. zo slova "slovo" sa stane "ovols",
- RSHIFT - posuň znaky A-Za-z na vstupe o jednu pozíciu "doprava" ako v cézarovej šifre, t.j. a->b, b->c, ..., y->z, z->a,
- LSHIFT - posuň znaky A-Za-z na vstupe o jednu pozíciu "doľava" ako v cézarovej šifre, t.j. a->z, b->a, ... z->y,
- SQUASHWS - zlúč viaceré za sebou nasledujúce "biele znaky" (whitespace) do jedného znaku (medzery).

Ako bonus si môžete vymyslieť ďalšie operácie.

Jednotlivé podprocesy a vlákna a k nim pridelené operácie budú spúšťané na základe parametrov príkazového riadku. napr.:

*./pipeline LOWERCASE REVWORD SQUASHWS < moj_subor.txt*

spusti aplikáciu s podprocesmi:

P1 s operáciou LOWERCASE

P2 s operáciou REVWORD

P3 s operáciou SQUASHWS

ktoré budú spracúvať obsah súboru moj_subor.txt a výsledok sa vypíše na štandardný výstup. Môžete predpokladať, že vstup bude v textovej forme, t.j. bude obsahovať iba zobraziteľné znaky zo znakovej sady ASCII.

Bonus spracovanie UTF-8.

# Moje riešenie:

Zadanie som riešil pomocou vlákien. Komunikácia medzi jednotlivými vláknami prebieha cez rúry. Jednotlivé procesy, ktoré to má podporovať sú na začiatku zdrojového kódu ako funkcie, ktoré sa potom spúšťajú v paralelných vláknach.

Ako parameter vstupuje smerník na štruktúru, ktorá obsahuje popisovače na vstupnú rúru, odkiaľ bude čítať znaky a na výstupnú rúru, kam sa bude transformovaný výstup zapisovať.

V hlavnom programe sa najskôr spustí prvý proces čítanie zo štandardného vstupu, potom sa v cykle zaradia ďalšie procesy, medzi ktorými sa prepoja jednotlivé rúry. Tie náväzne komunikujú a nakoniec sa spustí proces zapisovania na štandardný výstup. Hlavný program čaká na tento výstup, kedy sa dokončí (t.j. toto vlákno).

Ukončenie programu závisí práve na ukončení čítania. Keď prvý proces (čítanie) dokončí čítanie, (t.j. narazí na koniec dát), tak zatvorí rúru ďalšiemu vláknu. Toto následné vlákno to zistí a zatvorí rúru ďalšiemu vláknu a takto sa kaskádovo zatvoria všetky rúry a ukončia vlákna. Ukončí sa aj posledné vlákno (zapisovanie) a tým sa aj ukončí program.

Avšak, užívateľ potrebuje aj nápovedu, čiže „ktoré procesy sú podporované?“ Do nápovedy sa teda dostaneme obyčajným vykonaním programu, bez vložených argumentov, kde sú vypísané všetky podporované procesy. Program funguje pre znaky sady ASCII.

Príklad spustenia programu:

*./pipeline CAPITALIZE SQUASHWS < test*

**./pipeline NÁZOV_PROCESU_1 NÁZOV_PROCESU_2 NÁZOV PROCESU_N < Vstupný súbor**

Týchto procesov môže byť ľubovoľný počet a môžu sa aj opakovať. Pre každý proces sa teda spustí osobitné vlákno a tieto vlákna sa prepoja rúrami.

# Záver:

Pri tejto semestrálnej práci som sa naučil správne používať vlákna a rúry a mal som možnosť vyskúšať si prácu s nimi. Projekt mi ukázal výhodu práce s vláknami, ktorá je nezanedbateľná a to prehľadná práca s nimi (prehľadný kód).

Avšak, podmienkou dobre fungujúceho programu, ktorý využíva vlákna a rúry je dobre navrhnutý projekt.