Tyto úkoly pochází z cvičení Jana Hadravy z Úvodu do UNIXu v letním semestru roku 2017/2018. Stránky cvičícího se nachází na adrese `https://kam.mff.cuni.cz/~had/`

1

Stáhněte si ze stránek cvičení archiv a rozbalte jej. Vytvoří se vám určitá adresářová struktura. Vaším úkolem bude ji pomocí shellu (konkrétně příkazů ls, cd, more) prozkoumat a napsat takovou posloupnost příkazů, která vytvoří přesně tu samou strukturu.

- Proveďte tedy nejprve následující příkazy:
  `wget https://kam.mff.cuni.cz/~had/vyuka/1718/unix/adresarova_struktura.tar.gz`
  `tar -xf adresarova_struktura.tar.gz`
  - První příkaz stáhne soubor `adresarova_struktura.tar.gz` do aktuálního pracovního adresáře.
  - Druhý příkaz pak tento soubor rozbalí do adresáře `adresarova_struktura`, který obsahuje nějaké podadresáře/soubory.

Do okénka níže pak napište ty příkazy, které vytvoří adresář `adresarova_struktura` a vše uvnitř podle zadaného vzoru.

2

Vymyslete příkaz, který:

- Přečte soubor `A.in` a vypíše z něj řádky 4-8 do souboru `A.out`.
- Přečte soubor `B.in` a vypíše jej do `B.out` s tím, že pouze první řádku napíše pozpátku, zbytek nechá.
- Přečte soubor `C.in` a vypíše z něj posledních 5 řádek do souboru `C.out`, ovšem v obráceném pořadí.
- Přečte soubor `D.in` a vypíše jej do `D.out` v obráceném pořadí řádek. Přitom soubor zakódujte pomocí rot-13. To znamená, že znaky anglické abecedy posunou o 13 znaků a → n, b → o, ... m → z, n → a, o → b, p → c ... z → m. Nezapomeňte kódovat malá i velká písmena. Diakritiku řešit nemusíte.

3

Napište skript, který přečte vstupní data obsahující informace o pohybu na bankovním účtu a vypíše zůstatky po každé provedené operaci.

Skript bude zavolán s jedním až třemi parametry. První parametr bude obsahovat počáteční zůstatek, druhý název vstupního souboru, třetí název výstupního souboru. Pokud dostanete méně parametrů, tak místo chybějícího souboru použijte standardní výstup/vstup. Speciálně pokud bude skript zavolán se dvěma parametry, pak je první počáteční zůstatek, druhý vstupní soubor a výstup máte vypsat na standardní výstup.

Formát vstupu je následující, na každém řádku se nacházejí následující středníkem oddělené informace:

1. Popis transkace – textový popis typu transakce (neobsahuje středník)
2. Relativní částka – celé číslo (v Kč), kladné znamená přírůstek na účtu, záporné odchozí platbu
3. Popis pro příjemce – libovolný text neobsahující středník

Formát výstupu je obdobný, opět jsou na každéĺ řádku středníkem oddělené informace:

1. Relativní částka
2. Zůstatek po provedení transakce – jediná nově spočtená informace
3. Popis pro příjemce
4. Popis transkace

Případné pomocné soubory ukládejte do dočasného adresáře, který po sobě zase snažete. Udělejte to tak, aby se dalo jméno adresáře změnit tak, že stačí upravit jen jedno místo ve vašem kódu. Pokud dočasný adresář nejde z nějakého důvodu vytvořit, neprovádějte nic a vypište o tom chybovou hlášku.

4

Napište skript, který bude hledat v textových souborech zadaná slova a na výstupu vám bude říkat jejich pozice – jméno souboru, číslo řádku a pořadí slova na řádce.

Skript může být volán s následujícími parametry:

- `-i vstupní-soubor` – Tímto řekneme skriptu, že se má hledat v daném vstupním souboru. Tento argument se může opakovat. Potom skript bude hledat ve všech takto zadaných souborech.
- `-w slovo` – Tímto parametrem řekneme, jaké slovo se má hledat. Stejně jako u vstupních souborů, může být hledaných slov zadaných také více.
- `-w slovo` – Tímto parametrem řekneme, jaké slovo se má hledat. Stejně jako u vstupních souborů, může být hledaných slov zadaných také více.

- `-p část-slova` – Tímto parametrem řekneme, co musí slovo obsahovat, abychom vypsali jeho výskyt. Parametr se může opakovat a může být libovolně kombinovaný s ostatními parametry.
- `-P část-slova` – Stejné jako malé p, ale ignorujte velikost písmen.
- `-d znaky-považované-za-oddělovače` – Tímto parametrem předefinujeme, které znaky nepovažujeme za součást slova. Pokud není parametr zadaný, pak jako oddělovač berte pouze mezeru.

Formát výstupu je následující:
`název-vstupního-souboru:číslo-řádky:číslo-slova-na-řádce:nalezené-slovo`. Výstup vypisujte na standardní výstup. Pokud byste jeden výskyt slova měli vypsat díky více vyhledávacím pravidlům, může se stejný řádek ve výstupu objevit vícekrát. Na pořadí řádků ve výstupu nezáleží.

5

Napište skript, který bude nahrazovat slova podle zadaných pravidel. Dostanete dva argumenty: První je soubor, který máte přečíst, aplikovat na něj pravidla a změněnou verzi vypsat na standardní výstup. Druhý argument je soubor s pravidly, která máte na ten první použít.

První soubor je jen obyčejný textový soubor, není na něm nic speciálního.

Druhý soubor – soubor s pravidly je speciálnější. Na každém řádku se v něm nachází jedno pravidlo v jednom z následujících formátů:

```
najdi --> nahrad
nahrad <-- najdi
najdi -42-> nahrad
nahrad <-3- najdi
```

- Vaším úkolem je následně v prvním souboru nahradit slovo `najdi` za `nahrad`.
- Pokud je šipka jednoduchá (tj. `<--` nebo `-->`), tak máte danou náhradu provádět v celém souboru (máte nahradit každý výskyt slova najdi).
- Pokud šipka obsahuje nějaké číslo, tak se dané pravidlo týká pouze řádky z daným číslem a na jiné řádky se nevztahuje. (Takže máte nahradit všechny výskyty tohoto slova na jedné konkrétní řádce.)
- Mezi verzemi zleva doprava a zprava doleva není žádný významový rozdíl. Mají se chovat zcela totožně (jen se liší pořadí slov). (Není proto tak velký neodstatek, když váš skript bude podporovat jen jeden směr pravidel.)

- Můžete předpokládat, že slova neobsahují žádné speciální znaky.
- Pravidla se mají aplikovat přesně v tom pořadí, v jakém jsou uvedeny ve druhém souboru. Je tedy možné že jedno slovo nahradíte za jiné a následně jiným pravidlem za nějaké další slovo.
- Nemusíte speciálně řešit situace, kdy jedno slovo je potřetězcem jiného. Pokud máte pravidlo `srot --> bourak` a v textu se objeví slovo `kovosrot`, potom je správným výstupem `kovobourak`.
- Používejte `sed`. (Můžete použít i nějaké další příkazy, jako třeba `rm`, abyste po sobě uklidili pomocné soubory. Tyto příkazy by se ale neměly podílet na samotné podstatě úlohy. Takže například nepoužívejte cykly, a příkazy jako `cut` a `paste`.)

6

Dnešní úkol bude trochu uměle omezený. Abyste získali plný počet bodů, musíte tato omezení dodržet. Omezení jsou tentokrát velmi důležitá, nezapomeňte na ně! Aby to bylo zajímavější, máte na výběr ze dvou možností.

Napište funkci, která dostane soubory ve stejném formátu jako /etc/passwd a /etc/group a jméno nějaké skupiny. Na standardní výstup dané funkce vypište skutečná jména (jméno a příjmení) všech uživatelů, kteří do této skupiny spadají. Nezapomeňte, že každý uživatel má také primární skupinu, do které spadá. O tomto členství není uveden explicitní záznam v group, ale dozvíte se jej pouze z UID a GID v souboru passwd.

Formát vstupu: Funkce bude číst jediný klasický argument – jméno hledané skupiny. Cesty k souborům passwd a group dostane funkce předané pomocí proměnných `"$PASSWD"` a `"$GROUP"`.

1. varianta omezení: Nepoužívejte cykly, podmínky (kromě případného ošetřování chybových stavů), grep, sed ani awk. Naopak velmi doporučuji příkazy cut, sort, tr a join.

2. varianta omezení: Používejte výhradně sed – ale pozor, smíte jej spustit jen jednou, tj. v této variantě máte naopak zakázané cykly, podmínky (kromě případného ošetřování chybových stavů), grep, awk, cut, paste, tr, join, ... a další podobné příkazy. Také nemůžete použít trik z jednoho z předchozích úkolů (vygenerovat pomocí sedu skript pro sed).

Pozn.: V mém řešení byla použita první varianta.

7

Tentokrát je domácí úkol na příkaz `awk`, takže nepoužívejte žádný jiný shellový příkaz. (A to ani obyčejné `tr` na nějaké předzpracování vstupu.)

Na standardním vstupu dostanete nejprve neorientovaný graf zadaný pomocí matice sousednosti. Dále se budou vyskytovat jednotlivé instrukce, které máte s grafem udělat. Graf si můžete udržovat celý v paměti (dvojrozměrném poli v awk). Instrukce (operace), které s grafem budeme dělat budou následující:

- Povinné instrukce:
  - `add(u,v)` Přidá hranu mezi vrcholy u a v
  - `delete(u,v)` Odebere hranu mezi vrcholy u a v
  - `neighbours(u)` Vypíše čárkami oddělený seznam sousedů vrcholu u
  - `exist(u,v)` Vypíše 1 nebo 0 podle toho, zda aktuálně existuje hrana mezi vrcholy u a v
- Volitelné (vyberte si alespoň jednu, lépe však obě):
  - `isolate(u)` Odebere všechny hrany vrcholu u (tj. udělá z vrcholu u izolovaný vrchol)
  - `degree(u)` Vypíše stupeň vrcholu u

Na konci vypište graf jako matici sousednosti (po provedení všech zadaných operací), čísla oddělujte mezerami.

Formát vstupu: První řádek bude obsahovat:

```
graph(n)
```

čímž se řekne, na kolika vrcholech bude graf definovaný. Vrcholy následně budou číslované od 1 do n (včetně).

Následovat bude n řádek, každá bude popisovat jeden řádek matice sousednosti. Pokud implementujete alespoň dvě volitelné operace, nemusíte vstup validovat a můžete předpokládat, že se shodují čísla na souřadnicích [u,v] a [v,u].

Zbylé řádky pak budou obsahovat instrukce.