1. Úloha IOS (2024)

Popis úlohy

Jako správní technologičtí vizionáři se i Vaše firma rozhodla vytvořit svou vlastní kryptoměnovou burzu, nazvanou XTF (eXtortion, Travesty, Fraud). Burza svým klientům umožňuje vkládat finanční prostředky ve formě kryptoměn i fiat měn; provádět směnu svých prostředků za měny jiné; úschovu prostředků a půjčování prostředků (tzv. *margin*).

Aby měli uživatelé přehled o svých účtech (a prostředcích na nich), musí Vaše burza vést záznamy o provedených transakcích všech uživatelů. Vaše účetnictví je však velmi nepřehledné a chaotické (tak, jak by to mělo být u každého správného startupu), a všechny záznamy jsou tak mezi sebou promíchané a uložené v jednom nebo více textových, příp. komprimovaných, souborech. Přes grafické rozhraní byste však měli každému uživateli zobrazovat pouze záznamy z jeho obchodního účtu. V rámci předzpracování dat pro výstup tak bude zapotřebí záznamy uživatelů filtrovat a třídit.

Pokud však chcete skutečně uspět na poli kryptoměnových *fintech* společností, musíte růst rychleji a více než Vaše konkurence. Abyste nalákali co nejvíce nových uživatelů na Vaši burzu, slíbili jste jim pohádkové zhodnocení jejich uložených prostředků pomocí Vaší sesterské investiční společnosti. Protože finanční regulace jsou pro zpátečníky a ne pro vizionáře Vašeho střihu, nic Vám nebrání uživatelům vykazovat fiktivní zisk z investice.

Přestože jinak zásadně dbáte na využívání těch nejnovějších a nejmodernějších technologií (správné *buzzwords* Vám zajistí příliv dalšího kapitálu od investorů), v tomto konkrétním případě skutečně dává větší smysl vyřešit problém pomocí shell skriptu a nikoliv natrénované neuronové sítě.

Specifikace rozhraní skriptu

JMÉNO

- `xtf` - skript pro předzpracování logů z Vaší kryptoměnové burzy.

POUŽITÍ

- `xtf [-h|--help] [FILTR] [PŘÍKAZ] UŽIVATEL LOG [LOG2 [...]`

VOLBY

- `PŘÍKAZ` může být jeden z:
  - `list` – výpis záznamů pro daného uživatele.
  - `list-currency` – výpis seřazeného *seznamu* vyskytujících se měn.
  - `status` – výpis skutečného *stavu účtu* seskupeného a seřazeného dle jednotlivých měn.
  - `profit` – výpis *stavu účtu* zákazníka se započítaným fiktivním výnosem.
- `FILTR` může být kombinace následujících:
  - `-a DATETIME` – after: jsou uvažovány pouze záznamy PO tomto datu a čase (bez něj). `DATETIME` je formátu `YYYY-MM-DD HH:MM:SS`.
  - `-b DATETIME` – before: jsou uvažovány pouze záznamy PŘED tímto datem a časem (bez něj).
  - `-c CURRENCY` – jsou uvažovány pouze záznamy odpovídající dané měně.
- `-h` a `--help` vypíšou nápovědu s krátkým popisem každého příkazu a přepínače.

POPIS

1. Skript filtruje, seskupuje a vypisuje záznamy z logů kryptoměnové burzy. Pokud je skriptu zadán příkaz, nad filtrovanými záznamy daný příkaz provede.
2. Pokud má skript vypsat *seznam* nebo *stav účtu*, každá položka je vypsána na jeden řádek a pouze jednou. Pořadí řádků je dáno abecedně dle kódů měn. Položky se nesmí opakovat. Formát výstupu u *stavu účtu* pro každou měnu je `KOD : HODNOTA`.
3. `list` opisuje (filtrované) záznamy odpovídající uživateli `UZIVATEL` na standardní výstup v pořadí, v jakém se vyskytují ve vstupních log souborech.
4. `list-currency` vypisuje seznam kódů měn vyskytujících se v (filtrovaných) záznamech uživatele `UZIVATEL`. Kódy měn jsou seřazeny abecedně.
5. `status` spočítá a vypíše stav účtu pro každou měnu uživatele `UZIVATEL` na základě (filtrovaných) záznamů. Seznam měn a jejich zůstatků je seřazen abecedně dle kódů měn. V tomto případě nezahrnujte do výsledků fiktivní výnos z investic.
6. `profit` je podobný příkazu `status` s tím rozdílem, že **kladné** zůstatky měn jsou změněny navýšeny **o X %**, kde **X** je hodnota **proměnné prostředí** `XTF_PROFIT`. Jste však trochu zapomnětliví a občas tak zapomenete proměnnou nastavit. Pokud není proměnná prostředí `XTF_PROFIT` nastavena, uvažujte výchozí hodnotu 20 % (příliš vysoké zhodnocení by mohlo přitáhnout nechtěnou pozornost regulačních úřadů). Hodnota proměnné `XTF_PROFIT` bude vždy pouze nezáporné celé číslo.

DETAILY

1. Pokud skript nedostane na vstupu žádný příkaz, uvažujte výchozí příkaz `list`.
2. Skript umí zpracovat i logy komprimované pomocí nástroje `gzip` (v případě, že název log souboru končí `.gz`). Je zřejmé, že bez komprimace by bylo obtížné uchovávat velké množství dat na Vašich serverech.
3. Předpokládejte, že vstupní soubory nemohou mít jména odpovídající některému příkazu, přepínači nebo kódu měny.
4. Předpokládejte, že hodnota vstupního parametru `UZIVATEL` není shodná s hodnotou žádného příkazu, přepínače nebo kódu měny.
5. V případě uvedení přepínače `-h` nebo `--help` se vždy pouze vypíše nápověda a skript skončí (tedy, pokud by za přepínačem následoval nějaký příkaz nebo soubor, neprovede se).

Vstupní log soubory

1. Skript analyzuje záznamy (logy) pouze ze zadaných souborů v daném pořadí.
2. Formát logu je CSV, kde oddělovačem je znak středníku `;`. Formát je řádkový, každý řádek odpovídá záznamu transakce provedené na kryptoměnové burze ve tvaru

   ```
   JMENO UZIVATELE;DATUM A CAS;MENA;HODNOTA
   ```

   kde

   - `JMENO UZIVATELE` označuje uživatelské jméno klienta. Pro jednoduchost uvažujte, že se jedná o řetězec obsahující pouze tisknutelné ASCII znaky bez bílých znaků nebo znaku středníku.
   - `DATUM A CAS` jsou ve formátu `YYYY-MM-DD HH:MM:SS`
   - `MENA` je třípísmenný kód (krypto)měny, např `USD`, `EUR`, `CZK`, `BTC`, `ETH` atd.
   - `HODNOTA` je hodnota transakce. Záporná hodnota značí prodej (resp. výběr) a kladná hodnota značí nákup (resp. vložení) měny. `HODNOTA` je desetinné číslo s přesností na čtyři desetinná místa s tečkou jako oddělovacím znakem.

   Příklad záznamů:

   ```
   CryptoWiz;2024-01-14 23:43:13;EUR;-3000.0000
   CryptoWiz;2024-01-14 23:43:15;ETH;1.2313
   ```

   - První záznam značí prodej 3000 jednotek měny Euro (kód měny `EUR`) uživatelem `CryptoWiz`.
   - Druhý záznam značí nákup 1.2313 jednotek měny Ethereum (kód měny `ETH`), opět uživatelem `CryptoWiz`.
3. Můžete předpokládat, že záznamy jsou ve vstupních souborech uvedeny chronologicky a je-li na vstupu více souborů, jejich pořadí je také chronologické.
4. Nové kryptoměny vznikají téměř každý den a kontrola správnosti kódů měn by akorát brzdila Vaši schopnost rychlé inovace. Předpokládejte, že všechny vyskytující se kódy měn jsou validní a existující (není potřeba kontrolovat).
5. Nemusíte kontrolovat korektnost operací vzhledem ke kontextu ostatních operací, např. zda se jedná o korektní směnu měn nebo vložení prostředků. V nejhorším případě můžete vždy svést chybu na Vaše technicky neschopné uživatele.
6. Nemusíte kontrolovat korektnost operací vzhledem ke stavu účtu uživatele, např. zda má uživatel na účtu dostatek prostředků na nákup nebo prodej měny (můžete uvažovat, že uživatel má pro tyto operace k dispozici dostatečný *margin*).

Návratová hodnota

- Skript vrací úspěch v případě úspěšné operace. Interní chyba skriptu nebo chybné argumenty budou doprovázeny chybovým hlášením na standardní chybový výstup a neúspěšným návratovým kódem.

Implementační detaily

1. Skript žádný soubor nemodifikuje. Skript nesmí používat dočasné soubory. Povoleny jsou však dočasné soubory nepřímo tvořené jinými příkazy (např. příkazem `sed -i`).
2. Skript by měl mít v celém běhu nastaveno `POSIXLY_CORRECT=yes`.
3. Skript by měl běžet na všech běžných shellech (`dash`, `ksh`, `bash`). Pokud použijete vlastnost specifickou pro nějaký shell, uveďte to pomocí direktivy interpretu na prvním řádku souboru, např. `#!/bin/bash` nebo `#!/usr/bin/env bash` pro `bash`. Můžete použít GNU rozšíření pro `sed` či `awk`. Jazyky Perl, Python, Ruby, atd. povoleny nejsou.

   UPOZORNĚNÍ: některé servery, např. `merlin.fit.vutbr.cz`, mají symlink `/bin/sh -> bash`. Ověřte si proto, že skript skutečně testujete daným shellem. Doporučuji ověřit správnou funkčnost pomocí virtuálního stroje níže.
4. Skript musí běžet na běžně dostupných OS GNU/Linux, BSD a MacOS. Virtuální stroj, na kterém lze ověřit správnou funkčnost projektu, je k dispozici zde (login: `mantic`, heslo: `mantic`).
5. Čísla vypisujte v desítkovém zápisu **[ÚPRAVA 18.2.] oříznutá (tedy nemusíte zaokrouhlovat)** na čtyři desetinná místa. Pozor, některé nástroje (např. `awk`) mohou vědět čísla vypisovat implicitně pomocí vědeckého zápisu.

Odevzdání projektu

Odevzdávejte pouze skript `xtf` (nebalte ho do žádného archivu). Odevzdejte do IS, termín Projekt 1.

Rady

- Dobrá dekompozice problému na podproblémy Vám může značně ulehčit práci a předejít chybám.
- Naučte se *dobře* používat funkce v shellu.
- Nedoporučujeme řešit projekt výhradně pomocí generativní AI. Generování velkých úseků kódu (nebo rovnou celého projektu) může vést k podezření z plagiátorství v případě vysoké shody vygenerovaného kódu napříč studenty.
  - Studenti podezřelí z plagiátorství mohou být pozváni na podání vysvětlení a demonstraci, že svému kódu skutečně rozumí. V případě neúspěšné obhajoby hrozí, že na Vaše řešení bude pohlíženo jako na plagiát se všemi důsledky (tj. disciplinární řízení atp.)!
  - Použití AI jako asistenta (rádce, inteligentní debugger, návrh dekompozice, brainstorming) k vyřešení projektu je v pořádku. Ve vlastním zájmu se však raději vyvarujte přebírání velkých úseků kódu.

Možná rozšíření

Implementaci volitelných rozšíření můžete kompenzovat případnou ztrátu bodů v jiné části projektu (lze implementovat libovolný počet rozšíření). Rozšíření jsou nepovinná a plný počet bodů je možné získat i bez jejich implementace.

- Podpora specifikace parametrů a přepínačů (`FILTR`, `PŘÍKAZ`) a `UZIVATEL` v libovolném pořadí. Log soubor nebo soubory budou vždy zadány jako poslední.
- Filtr `-c` je možné specifikovat vícenásobně. Příkazy `list`, `list-currency`, `status` a `profit` budou následně uvažovat ty záznamy, které obsahují některou z uvedených měn.

Příklady použití

- Ukázky vstupních log souborů jsou dostupné zde: https://pajda.fit.vutbr.cz/ios/ios-24-1-logs.

Příklady:

```
$ cat cryptoexchange.log
Trader1;2024-01-15 15:30:42;EUR;-2000.0000
Trader2;2024-01-15 15:31:12;BTC;-9.8734
Trader1;2024-01-16 18:06:32;USD;-3000.0000
CryptoWiz;2024-01-17 08:58:09;CZK;10000.0000
Trader1;2024-01-20 11:43:02;ETH;1.9417
Trader1;2024-01-22 09:17:40;ETH;10.9537

$ ./xtf Trader1 cryptoexchange.log
Trader1;2024-01-15 15:30:42;EUR;-2000.0000
Trader1;2024-01-16 18:06:32;USD;-3000.0000
Trader1;2024-01-20 11:43:02;ETH;1.9417
Trader1;2024-01-22 09:17:40;ETH;10.9537

$ ./xtf list Trader1 cryptoexchange.log
Trader1;2024-01-15 15:30:42;EUR;-2000.0000
Trader1;2024-01-16 18:06:32;USD;-3000.0000
Trader1;2024-01-20 11:43:02;ETH;1.9417
Trader1;2024-01-22 09:17:40;ETH;10.9537

$ ./xtf -c ETH Trader1 cryptoexchange.log
Trader1;2024-01-20 11:43:02;ETH;1.9417
Trader1;2024-01-22 09:17:40;ETH;10.9537

$ ./xtf -c GBP Trader1 cryptoexchange.log
$

$ ./xtf list-currency Trader1 cryptoexchange.log
ETH
EUR
USD

$ ./xtf status Trader1 cryptoexchange.log
ETH : 12.8954
EUR : -2000.0000
USD : -3000.0000

$ ./xtf -b "2024-01-22 09:17:40" status Trader1 cryptoexchange.log
ETH : 1.9417
EUR : -2000.0000
USD : -3000.0000

$ ./xtf -a "2024-01-15 16:00:00" -b "2024-01-22 09:17:41" status Trader1 cryptoexchange.log
ETH : 12.8954
USD : -3000.0000

$ ./xtf profit Trader1 cryptoexchange.log
ETH : 15.4744
EUR : -2000.0000
USD : -3000.0000

export XTF_PROFIT=40
$ ./xtf profit Trader1 cryptoexchange.log
ETH : 18.0535
EUR : -2000.0000
USD : -3000.0000
```

Příklad s více logy:

```
$ cat cryptoexchange-1.log
Trader1;2024-01-15 15:30:42;EUR;-2000.0000
Trader2;2024-01-15 15:31:12;BTC;-9.8734
Trader1;2024-01-16 18:06:32;USD;-3000.0000
$
$ gunzip -ck cryptoexchange-2.log.gz
CryptoWiz;2024-01-17 08:58:09;CZK;10000.0000
Trader1;2024-01-20 11:43:02;ETH;1.9417
Trader1;2024-01-22 09:17:40;ETH;10.9537
$
$ ./xtf status Trader1 cryptoexchange-1.log cryptoexchange-2.log.gz
ETH : 12.8954
EUR : -2000.0000
USD : -3000.0000
```

Příklady rozšíření:

```
$ ./xtf Trader1 status -a "2024-01-15 16:00:00" -b "2024-01-22 09:17:41" cryptoexchange.log
ETH : 12.8954
USD : -3000.0000

$ ./xtf -c ETH -c USD Trader1 cryptoexchange.log
Trader1;2024-01-16 18:06:32;USD;-3000.0000
Trader1;2024-01-20 11:43:02;ETH;1.9417
Trader1;2024-01-22 09:17:40;ETH;10.9537

$ ./xtf -c ETH -c EUR -c GBP list-currency Trader1 cryptoexchange.log
ETH
EUR
```


- Popis úlohy
- Specifikace rozhraní skriptu
- Vstupní log soubory
- Návratová hodnota
- Implementační detaily
- Odevzdání projektu
- Rady
- Možná rozšíření
- Příklady použití