Практикум из рачунарских алата

Вежбе 2

Исидора Грујић
isidora@uni.kg.ac.rs

Лазар Илић
lazar@uni.kg.ac.rs

Филип Милић
milicf@uni.kg.ac.rs

Катедра за електротехнику и рачунарство
Факултет инжењерских наука Универзитета у Крагујевцу

Крагујевац, 29. октобар 2024.

Садржај

Садржај

Употреба џокера

Већ је наговештена експанзија команди употребом посебних карактера - *џокера*, којим се врло ефикасно може навести рачунар да самостално прати одређени образац. Могуће је позив више (стотина, хиљада,...) команди свести на само 1.

Џокери се конкретно користе како би се убрзала манипулација именима фајлова који се обрађују, конкретно, генерише се калуп/маска којим се врши селекција фајлова од интереса. Да би фајл био *изабран* неопходно је да се његов назив уклопи у образац дефинисан џокерима.

Употреба џокера

Наравно, специјалних карактера има пуно, и неки карактери су специјални на више начина, односно, њихово значење варира зависно од случаја употребе. Списак џокера (као и неколико примера употребе истих) је дат у поглављу 4 у оквиру TLCL, а пар најчешће виђених примера излистан је испод:

- * - може заменити било коју секвенцу карактера, односно произвољан карактер који се појављује произвољно много пута;
- ? - мења било који карактер (дакле карактер који се појављује тачно једном);
- [set] - проверава да ли карактер припада одређеном скупу. Скуп (*set*) може бити дефинисан од стране корисника, или пак један од предефинисаних скупова.

Додатни специјални карактери

Поред џокера, ваља поменути и употребу витичастих заграда за генерисање образаца које прати командна линија. Њихова употреба омогућава упрошћавање понављајућих задатака, рецимо нумерисање фајлова на основу познатог опсега бројева.

Неопходно је и поменути $ као специјални карактер. Долар се користи за дохватање некакве вредности, и његово понашање зависи од начина позивања:

- $VAR - узима вредност променљиве VAR,
- $(command) - узима резултат, односно повратну вредност позива команде command,
- $((izraz)) - евалуира и узима вредност izraz-a.

За додатна појашњења и примере, погледати поглавље 7 TLCL. Сасвим је разумно поставити питање: *„А шта ако је потребно посматрати до сада помињане карактере без осврtања на њихове специјалне функционалности?"* О такозваном *escape*-овању карактера такође се можете информисати унутар поглавља 7.

Реглурани изрази - regex

Најпростије речено, регуларни изрази представљају начин да рачунару представимо образац који ће користити за претрагу садржаја од интереса. Превасходно се користе за манипулацију текстуалног садржаја, и интегрисани су у функцију *grep*, која се користи за претрагу садржаја фајлова. Могуће је регуларне изразе користити и за претрагу назива и путања до фајлова, у комбинацији са функцијом *find*.

Листа карактера са специјалним значењем унутар регуларних израза дата је испод:

```
^ $ . [ ] { } - ? * + ( ) | \
```

Погледати поглавље 19 TLCL.

Садржај

Мотивација

Рачунари сами по себи нису нешто много паметни, али су изузтено брзи. Пун потенцијал рачунара подразумева оптимално писање програма с поменутим перформансама на уму. Из перспективе инжењера/програмера, неопходно је покушати пребацити што виши степен оптерећења на рачунар.
Познати су концепти као што су писање функција за блокове кода који се више пута позивају приликом извршавања неког задатка, као и технике контрола тока гранањем или петљама (поглавља 27, 29, 31, 33 TLCL).
Додатан ниво апстракције уводе скрипте. Скрипте су у основи фајлови састојани из низа команди. Љуска (*shell*) може прочитати скрипту и извршити уписане команде, као да су редом унесене кроз интерфејс командне линије. Иако изискују времена и труда иницијално да би се адекватно направиле и биле издебагиране, уколико се њихова функционалност користи више пута, на дугорочном плану штеде кориснику много времена.

Смернице за скриптовање

- Нагласити који интерпретатор извршава скрипту, конкретно за пример bash скрипти следећом линијом кода: `#!/bin/bash`,
- Провера путање до *bash* интерпретатора са `$which bash`
- Да би shell видео скрипту као извршни фајл, подесити одговарајуће привилегије за извршавање (употребом команде *chmod*),
- Размотрити додавање путање скрипте у $PATH променљиву,
- Кориснички унос? Постоје 2 опције:
  - Унутар скрипте - команда *read*, пандан команди *input()* у Python-у (поглавље 28 TLCL)
  - Приликом позива скрипте, односно преузимање аргумената из командне линије (поглавље поглавље 32 TLCL).

Садржај

Пример/Задатак 1

Написати bash скрипту која захтева од корисника назив директоријума који ће бити креиран. Уколико директоријум са тим називом већ постоји, обуставити извршавање скрипте.
Потом је потребно приступити новонасталом директоријуму, као и иштампати путању на којој се он налази. Затим, направити 4 празна фајла произвољног назива, проверити резултат позивом команде *ls*.
У генерисане фајлове исписати произвољан садржај, а онда их све ишчитати, односно уписани садржај исписати у терминалу. За крај, почистити за собом, односно вратити се у изворни директоријум, и обрисати све креиране фајлове и поддиректоријум.

Пример/Задатак 2

Написати bash скрипту унутар које су дефинисане 3 произвољне функције.
Скрипта по позиву очекује кориснички унос у виду једног од бројева 1, 2, или 3; а затим позива одговарајућу функцију.

Пример/Задатак 3

Написати bash скрипту која очекује тачно 1 параметар из командне линије. Подразумевано је прослеђена вредност параметра (аргумент при позиву функције) број између 1 и 12, а задатак скрипте је да на основу унетог редног броја у конзолу иштампа одговарајући месец у години.

Пример/Задатак 4

Написати скрипту којој се прослеђује тачно један аргумент са командне линије. Исписати аргумент уколико је прослеђен тачно један, а у супротном обавестити корисника о неадекватном коришћењу скрипте.