

Vaardigheidsproef scripting - examenversie

Academiejaar	2023-2024
Datum Examen	4 september 2024
Opleiding	Toegepaste informatica, Jaar 2
Keuzepakket	Data Engineering
Opleidingsonderdeel	Linux for Data Scientists - Vaardigheidsproef
Lesgever(s)	Bert Van Vreckem, Thomas Parmentier

Tijdens dit deel van het examen mogen dezelfde hulpmiddelen gebruikt worden als het schriftelijk deel.

Algemene richtlijnen

Deze opgave is een variant op de de take-home opdracht die je in principe al uitgewerkt hebt. In deze opgave zijn een aantal wijzigingen aangebracht, deze worden in het **groen** weergegeven.

Je start met je uitgewerkte oplossing van de take-home opgave en maakt de nodige wijzigingen. Hierop word je gequoteerd. Elke gevraagde wijziging heeft een label (**CR-nn**) (CR = change request, gevolgd door een nummer). Gebruik deze in de codecommentaar van je script om aan te geven waar je de wijziging hebt aangebracht.

Laad het aangepaste script op via exam.hogent.be.

Minimale requirements voor inhoudelijke beoordeling

De hieronder opgesomde criteria zijn noodzakelijk om een bestand als script te kunnen uitvoeren. Inzendingen die hier niet aan voldoen kunnen we dan ook niet inhoudelijk beoordelen en krijgen meteen 0:

- Het resultaat van zowel `bash -n task.sh` als `shellcheck --severity=error task.sh` moet succesvol zijn (dus zonder fouten).
- Het script mag geen DOS regeleindes (CRLF) hebben, anders kan Bash het niet interpreteren
- Het script moet een geldige "shebang" hebben op de eerste regel
- Als we het script uitvoeren met optie `help`, dan moet dit lukken (we krijgen dus de Usage: boodschap te zien en de exit-status is 0)

Pas als al deze criteria voldaan zijn, kunnen we ook verder inhoudelijk beoordelen!

We verwachten verder ook dat je gebruik maakt van het aangeleverde sjabloon en de daarin gedefinieerde functies implementeert. Je mag uiteraard wel extra functies toevoegen als je dat nuttig vindt.

Opgave

`task.sh` is een script dat een Linux gebruiker helpt bij het bijhouden van een todo-lijstje.

- Als je `help` opgeeft, drukt het script een hulpboodschap af en sluit meteen af met exit-status 0.
Gebruik een Here Document!

```
Usage: ./task.sh COMMAND [ARGUMENTS]...

add 'TASK DESCRIPTION'
    add a task
done ID
    mark task with ID as done
dump
    show all tasks, including task ID
edit
    edit the task file
list-contexts
    show all contexts (starting with @)
list-tags
    show all tags (starting with #)
overdue
    show all overdue tasks
search 'PATTERN'
    show all tasks matching (regex) PATTERN
```

TASK FORMAT

A task description is a string that may contain the following elements:

- @context, i.e. a place or situation in which the task can be performed (See Getting Things Done) e.g. @home, @campus, @phone, @store, ...
- #tag tags, that can be used to group tasks within projects, priorities, etc. e.g. #linux, #de-prj, #mit (most important task), ... Multiple tags are allowed!
- yyyy-mm-dd a due date in ISO-8601 format

In the task file, each task will additionally be assigned a unique ID, an incrementing integer starting at 1.

Algemene requirements

- Zorg dat je naam en emailadres vermeld zijn op de voorziene plaats in de commentaar bovenaan het script!
- Gebruik shell-opties om de robuustheid van het script te verhogen (bv. behandelen van onbestaande variabelen).
- Gebruik ShellCheck om fouten te voorkomen!
- Gebruik `stdout` exclusief voor het afdrukken van informatie uit het takenbestand: taakbeschrijvingen, contexten, labels, enz. Foutboodschappen, waarschuwingen, enz. worden afgedrukt op `stderr`.
- Vermijd "hard-coded" waarden in de code, gebruik zoveel mogelijk variabelen!
- Het script wordt altijd opgeroepen met als eerste argument een commando. Als dit niet het geval is, wordt verondersteld dat de gebruiker `help` bedoelde.

- **CR-01:** Zorg ervoor dat de commando's niet hoofdlettergevoelig zijn. Zo moeten bijvoorbeeld ook **ADD** of **Add** in plaats van **add** en **EDIT** of **EdIT** in plaats van **edit** aanvaard worden.

```
$ ./task.sh d
Ambiguous command: d. Did you mean 'done' (do) or 'dump' (du)?
$ ./task.sh l
Ambiguous command: l. Did you mean 'list-contexts' (lc) or 'list-tags' (lt)?
```

Instellingen

Instellingen die de gebruiker kan aanpassen worden opgeslagen in een configuratiebestand `~/.taskrc`. De instellingen worden ingelezen met **source** en zijn dus Bash-syntax.

Als het bestand niet gevonden wordt, wordt het aangemaakt met standaardwaarden.

Volgende instellingen zijn mogelijk:

Instelling	Standaardwaarde
TASK_FILE	Bestand .tasks in de home-directory van de gebruiker
TASK_EDITOR	Absoluut pad naar nano
TASK_ARCHIVE	CR-02: Bestand .task-archive in de home-dir van de gebruiker.

Voorbeeld:

```
$ ./task.sh help
Settings file not found. Creating it...
Done. Please edit '/home/linuxmint/.taskrc' if you want to change settings.
Usage: ./task.sh COMMAND [ARGUMENTS]...

add "TASK"
...
$ cat ~/.taskrc
# task.sh settings file
TASK_FILE=~/.tasks"
TASK_ARCHIVE=~/.task-archive"
TASK_EDITOR=/usr/bin/nano
```

Taak toevoegen

Met **add** kan je een nieuwe taak toevoegen. Elke taak krijgt een ID, een geheel getal beginnend bij 1 (zie verder). Na toevoegen van de taak wordt deze ID afgedrukt. De taak wordt toegevoegd aan het einde van het taakbestand. Aan het begin van de lijn komt het ID, gevolgd door een TAB-karakter en vervolgens de taakbeschrijving zelf.

Voorbeeld:

```
$ ./task.sh add 'Buy sugar, milk, eggs, flour for #pancakes @store'
Added task 1
$ ./task.sh add
Missing task description!
```

Als je geen taakbeschrijving opgeeft, stopt het script met een geschikte foutmelding en exit-status.

Het script bevat een functie `get_next_task_id` die de laagst mogelijke ID-waarde teruggeeft die nog niet in gebruik is. Deze wordt bepaald door in het takenbestand te zoeken naar reeds gebruikte IDs, beginnend met 1, vervolgens 2, enz. totdat een vrij ID gevonden wordt. Deze wordt dan toegekend aan de nieuwe taak.

In de beschrijving van een taak kan je volgende elementen gebruiken:

- Een deadline, in de vorm van `jjjj-mm-dd`. **CR-03: Dit is verplicht voor elke taak.**
- Een "context", in de vorm van `@context`. Hiermee bedoelen we de plaats waar de taak kan uitgevoerd worden (bv. `@home`, `@campus`, `@phone`, ...). Dit is een concept uit het productiviteitssysteem [Getting Things Done](#).
- Een "tag", in de vorm van `#tag`. Dit kan de naam van een project zijn, een trefwoord, prioriteitcode, ... Meerdere tags zijn mogelijk. Let op! Omdat Bash het `#`-teken speciaal behandelt (commentaar), moet je dit op de CLI escaperen met een `\`-teken of de taakbeschrijving tussen aanhalingstekens zetten.
- **CR-04: Je kan geen twee keer dezelfde taak toevoegen, d.w.z. een taak met een volledig identieke beschrijving als een bestaande taak. In dit geval toon je een gepaste foutboodschap**

Voorbeeld:

```
$ ./task.sh add 'Buy sugar, milk, eggs, flour for #pancakes @store 2024-09-15'
Added task 1
$ ./task.sh add 'Bake #pancakes @home 2024-09-16'
Added task 2
$ ./task.sh add 'Bake #pancakes @home 2024-09-16'
Task was not added: duplicate detected
$ ./task.sh add Eat all the \#pancakes @home 2024-09-19
Added task 3
$ ./task.sh add Do the dishes \#pancakes @home
Missing deadline, task not added.
$ ./task.sh add Do the dishes \#pancakes @home 2024-09-22
Added task 4
```

Taak als afgerond markeren

Met `done` kan je een taak als afgerond markeren. Dit doe je door het ID van de taak op te geven. Het script toont de taak en vraagt bevestiging. **CR-05: Als de gebruiker antwoordt met 'y', wordt de taak uit het taakbestand verwijderd en verplaatst naar het task-archive in de home-directory van de gebruiker, zo niet blijft de taak ongewijzigd. Bij het archiveren van taken wordt het ID van de taak verwijderd, de rest van de omschrijving blijft ongewijzigd.**

Als de gebruiker geen ID opgeeft, of de ID komt niet voor in het takenbestand, stopt het script met een geschikte foutmelding en exit-status.

Voorbeeld:

```
$ ./task.sh done 6
Task: 6 Call mom @phone
Mark task 6 as done? [y/N] N
Task 6 kept
$ ./task.sh done 6
Task: 6 Call mom @phone
Mark task 6 as done? [y/N] y
Task 6 marked as done
$ ./task.sh done
Missing task ID!
$ ./task.sh done 999
Task 999 not found!
```

Alle taken afdrukken

Met `dump` druk je de inhoud van het takenbestand af.

Voorbeeld:

```
$ ./task.sh dump
1  Buy sugar, milk, eggs, flour for #pancakes @store
2  Bake #pancakes @home
3  Eat all the #pancakes @home
4  Do the dishes #pancakes @home
```

Taakbestand bewerken

Soms wil je wijzigingen aanbrengen aan een taak. Dit kan met `edit`. Het script opent het takenbestand in de teksteditor die in de instelling `TASK_EDITOR` staat.

```
$ ./task.sh edit
[... editor opent ...]
```

Overzicht contexten

Met `list-contexts` druk je een alfabetisch gesorteerd overzicht af van alle contexten die in het takenbestand voorkomen, met voor elk het aantal taken. Dit zijn woorden die beginnen met een `@`-teken.

(CR-06) De contexten worden weergegeven volgens dalende hoeveelheid taken.

Voorbeeld:

```
$ ./task.sh list-contexts
5 @campus
10 @home
2 @phone
4 @store
```

Overzicht tags

Met `list-tags` druk je een gesorteerd overzicht af van alle tags die in de taken voorkomen. Dit zijn woorden die beginnen met een `#`-teken. Let er op dat als een taakbeschrijving meerdere tags bevat, alle tags ook in de lijst voorkomen.

(CR-07) De tags worden weergegeven volgens dalende hoeveelheid taken.

Voorbeeld:

```
$ ./task.sh list-tags
#chores
#de-prj
#pancakes
#world-domination
```

Taken zoeken

Met `search` kan je zoeken naar taken die een bepaald (regex) tekstpatroon bevatten. Het script drukt alle taken af die voldoen aan het patroon, samen met hun ID.

Voorbeeld:

```
$ ./task.sh search pancakes
1 Buy sugar, milk, eggs, flour for #pancakes @store
2 Bake #pancakes @home
3 Eat all the #pancakes @home
4 Do the dishes #pancakes @home
$ ./task.sh search
Missing search pattern!
```

Verlopen taken

Met `overdue` toon je een lijst van taken met een deadline die verstreken is.

Voorbeeld:

```
$ ./task.sh overdue
1 Buy sugar, milk, eggs, flour for #pancakes @store 2023-02-02
```

```
6 Ask boss for a raise @work 2022-12-31
7 Update #world-domination plans @home 2023-01-01
```

Uitstellen

(CR-08) Met `postpone` kan je de deadline van een bepaalde taak uitstellen. Je geeft een taaknummer mee, samen met het aantal dagen waarmee de deadline opgeschoven moet worden. Als je geen termijn meegeeft, wordt de deadline standaard met 7 dagen opgeschoven.

Voorbeeld:

```
$ ./task.sh postpone 7 2
Task: 7 Update #world-domination plans 2024-09-14 @home
Postpone task 7 with 2 days? [y/N]
y
Task 7 postponed until 2024-09-16
```