fsh (free shell)

დავალების მიზანია შექმენათ პროგრამა **fsh (free shell)**. ის უნდა იყოს ‘unix shell’-ის ერთ ერთი იმპლემენტაცია მსგავსად ‘bash’-ის, ‘ksh’-ს და ‘csh’-სა. **fsh**-ს იმპლემენტაცია უნდა აკმაყოფილებდეს ამ დოკუმენტში მოცემულ ყველა მოთხოვნას.

fsh-ს უნდა ჰქონდეს ჩაშენებული (built-in) ფუნქციები:

- **?**-- ბეჭდავს ინფორმაციას **fsh**-ს შესახებ
- **cd** -- ცვლის დირექტორიას
  - იხილეთ **man 2 chdir**
- **pwd** -- ბეჭდავს მიმდინარე სამუშაო დირექტორიის სახელს
  - იხილეთ **man 2 getcwd**
- **exit** -- ასრულებს (თიშავს) შელის პროცესს
  - **exit k** ასრულებს პროცესს და მშობელს უბრუნებს **k** სტატუს კოდს
- **ulimit** -- აბრუნებს მიმდინარე ლიმიტებს ან ცვლის მათ
  - იხილეთ **man ulimit**
  - იხილეთ **man getrlimit**
  - იხილეთ **man setrlimit**
- **nice** -- უშვებს პროგრამას შეცვლილი **‘nice’**-ით
  - თუ **nice**-ს გამოვიძახებთ პარამეტრების გარეშე, დაბეჭდავს მიმდინარე პროცესის **niceness**-ს
  - იხილეთ **man 2 nice**
- **kill** -- გზავნის სიგნალს მითითებულ პროცესთან
  - იხილეთ **man 2 kill**
- **type** -- ბეჭდავს გადაცემული ბრძანება **built-in** ფუნქციაა თუ სხვა პროგრამა
  - თუ ჩაშენებული ფუნქციის და სხვა პროგრამის სახელები ემთხვევა, ყველა შედეგი უნდა დაიბეჭდოს სხვადასხვა ხაზზე
  - აღწერა
- **echo $VARNAME** -- ბეჭდავს მითითებული ცვლადის მნიშვნელობას
  - იხილეთ **man getenv**
- **echo $?** -- ბეჭდავს ბოლო შვილობილი პროცესის სტატუს კოდს
- **echo "some random string"** -- ბეჭდავს გადმოცემულ სტრინგს **stdout**-ზე
- **export VARIABLE** -- აექსპორტებს ცვლადს და მის მნიშვნელობას
  - იხილეთ **man setenv** ან/და **man putenv**

საკონტროლო ოპერატორები:

- **fsh** უნდა იძლეოდეს საკონტროლო ოპერატორების გამოყენების საშუალებას
  - (**‘||’** და **‘&&’**)
- თუ ბრძანებები გადაბმულია **‘||’** ოპერატორით მარჯვენა ოპერანდი სრულდება მაშინ როცა მარცხენა ოპერანდი შედეგად დააბრუნებს არანულოვან კოდს(მუშაობას დაასრულებს შეცდომით).

- თუ ბრძანებები გადაბმულია **‘&&’** ოპერატორით მარჯვენა ოპერანდი სრულდება მაშინ როცა მარცხენა ოპერანდი შედეგად დააბრუნებს **0**-ს(მუშაობას დაასრულებს წარმატებით).
- მაგ. ბრძანებები:
    - **ls none_existing_dir || echo "there is no such directory or file"**
    - **ls ./ && echo "ls executed successfully"**

‘პაიპ’ ოპერატორი:

- **fsh** უნდა იძლეოდეს **PIPE** ინტერპროცეს კომუნიკაციის მექანიზმის გამოყენების საშუალებას.
- ‘პაიპის’ აღმნიშვნელ ოპერატორად გამოიყენეთ შემდეგის სიმბოლო **‘|’**
- მაგ: ბრძანება **ls /etc | grep passwd | sort**
    - **ls**-ის **stdout**-ი უნდა გადაებას **grep**-ის **stdin**-ს
    - **grep**-ის **stdout** უნდა გადაებას **sort**-ის **stdin**-ს

I/O stream-ების გადამისმართება:

- **fsh**-ს უნდა შეეძლოს შვილობილი პროცესის **I/O stream**-ების გადამისამართება
- ოპერატორი **‘<’** ამისამართებს სტანდარტულ ინფუთს **(stdin)**
- ოპერატორი **‘>’** ამისამართებს სტანდარტულ აუთფუთს **(stdout)**
    - ჯერ ფაილის ზომას ანულებს
- ოპერატორი **‘>>’** ამისამართებს სტანდარტულ აუთფუთს **(stdout)**
    - ახალ მონაცემებს ბოლოში აწებებს
- მაგ: ბრძანება **cat fsh.c > fsh_temp.txt**
    - **fsh.c** ფაილის შიგთავსი გადააქვს **fsh_temp.txt** ფაილში
    - (**fsh_temp.txt** ფაილის ზომა დასაწყისში ნულდება)

შემოსული მონაცემების დამუშავება:

- **fsh** უნდა ინახავდეს შეყვანილი ბრძანებების ისტორიას.
- ისტორიაში ძებნა უნდა ხორციელდებოდეს სანავიგაციო ღილაკების გამოყენებით
    - (**Up**, **Down**) ღილაკებით.
- ინფუთის დასამუშავებლად შეგიძლიათ გამოიყენეთ **readline** ბიბლიოთეკა.
    - იხილეთ **man 3 readline**

დამატებითი მოთხოვნები:

- გარდა ჩაშენებული ბრძანებებისა **fsh**-ს უნდა შეეძლოს ნებისმიერი გარე პროგრამის გაშვება, რომლის გამშვები ფაილიც განთავსებულია **$PATH** ცვლადში მოცემული დირექტორიებიდან რომელიმეში.
- **fsh**-ს უნდა შეეძლოს მიმდინარე დირექტორიაში არსებული გამშვები “exe” ფაილების გაშვება
    - თუ ფაილის სახელია **program**, **./program** უნდა უშვებდეს მას.
- **fsh** პროცესი არ უნდა "კვდებოდეს" ღილაკების კომბინაციით **CTRL-C** ან **CTRL-Z**
    - იხილეთ **man 2 signal**

- **-c** პარამეტრით პროგრამა არგუმენტად უნდა იღებდეს შესასრულებელ ბრძანებას.
    - გადაცემული ბრძანების შესრულების შემდეგ უნდა ამთავრებდეს მუშაობას და აბრუნებდეს შესრულებული ბრძანების სტატუსს.
    - მაგ. ბრძანება: **./fsh -c 'cd /tmp; pwd'** უნდა ბეჭდავდეს **/tmp**-ს
    - სიმბოლო **';'** აღნიშნავს ბრძანების დასასრულს მას შეიძლება მოსდევდეს **N** რაოდენობის სხვა ბრძანება.

კარგი იქნება თუ...

- ყველა კარგ პროექტს მოყვება **README** ფაილი, რომელიც პროექტის ერთგვარ აღწერას წარმოადგენს. კარგი იქნება თუ ამ პროექტსაც მოაყოლებთ მას.
    - ერთ ერთი ვარიანტი იხილეთ აქ

!!! გაითვალისწინეთ !!!

პროექტი არ ჩაითვლება შესრულებლად, თუ არ აკმაყოფილებს შემდეგ მოთხოვნებს:

1. უნდა ახლდეს (მუშა) **makefile**-ი
2. უნდა კომპილირდებოდეს შეცდომებისა და warning-ების გარეშე.
3. საჭირო დამხმარე ბიბლიოთეკები უნდა დააყენოს **makefile**-მა
    - ან გამზადებული მოყვებოდეს კოდს.
4. მუშაობის პროცესში არ უნდა ითიშებოდეს, არ უნდა ხდებოდეს **segfault**-ი
    - გამოიყენეთ **valgrind**-ი გაგიმარტივებთ სამუშაოს.
5. მოცემულ ტესტებს უნდა გადიოდეს კოდის გადაკეთების გარეშე
6. ყოველი ფუნქცია კარგად უნდა იყოს დოკუმენტირებული.
    - თავზე კომენტარის სახით გარკვევით ეწეროს თუ რას აკეთებს.

პროექტი შემოწმდება ubuntu 14.04 64bit სისტემაზე.

დამხმარე მასალა:

- თავები წიგნიდან **The Linux Programming Interface:**
    - **TLPI თავი 3:** System Programming Concepts (ზოგადად გადახედეთ)
    - **TLPI თავი 4:** FILE I/O: The Universal I/O Model
    - **TLPI თავი 5:** FILE I/O: Further Details
    - **TLPI თავი 6:** Processes
- საჭირო **syscall**-ების ჩამონათვალი:
    - **open** **(TLPI: თავი 4.3)**
    - **close** **(TLPI: თავი 4.6)**
    - **dup2** **(TLPI: თავი 5.5)**
    - **truncate** **(TLPI: თავი 5.8)**
    - **setenv** **(TLPI: თავი 6.7)**
    - **getenv** **(TLPI: თავი 6.7)**
    - **putenv** **(TLPI: თავი 6.7)**
    - **chdir** **(TLPI: თავი 18.10)**
    - **getcwd** **(TLPI: თავი 18.10)**
    - **signal** **(TLPI: თავი 20.3)**

    - **kill** **(TLPI: თავი 20.5)**
    - **fork** **(TLPI: თავი 24)**
    - **exec** **(TLPI: თავი 27)**
    - **getrlimit** **(TLPI: თავი 36.2)**
    - **setrlimit** **(TLPI: თავი 36.2)**
    - **pipe** **(TLPI: თავი 44)**
    - **nice** **(man nice)**
- **man 2 syscalls** ბრძანება:
    - ბეჭდავს ყველა **syscall**-ის ჩამონათვალს
- **man 2 somename_syscall** ბრძანება:
    - ბეჭდავს კონკრეტული **syscall**-ის დოკუმენტაციას
    - მაგ: **man 2 exit** გიჩვენებთ **exit syscall**-ის დოკუმენტაციას