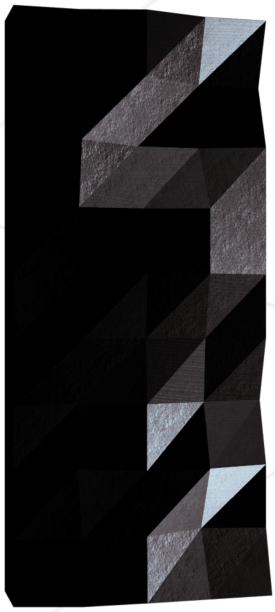

Pipex

Özet: Bu proje halihazırda bildiğiniz UNIX mekanizmasını programlayarak yapacağınız detaylı bir keşiftir.

Versiyon: 2

İçindekiler

Bölüm I

Önsöz

Cristina: `"Git bir yerde salsa dansı yap :)"`

Bölüm II

Genel Talimatlar

- Projeleriniz C programlama dilinde yazılmalıdır.
- Projeleriniz Norm'a uygun olarak yazılmalıdır. Bonus dosyalarınız/fonksiyonlarınız varsa, bunlar norm kontrolüne dahil edilir ve bu dosyalarda norm hatası varsa `0` alırsınız.
- Tanımlanmamış davranışlar dışında sizin fonksiyonlarınız beklenmedik bir şekilde sonlanmamalıdır (Segmentasyon hatası, bus hatası, double free hatası, vb.) . Eğer bunlar yaşanırsa s `0` alırsınız.
- Heap'de ayırmış olduğunuz hafıza adresleri gerekli olduğu durumlarda serbest bırakılmalıdır. Hiçbir istisna tolere edilmeyecektir.
- Eğer verilen görev `Makefile` dosyasının yüklenmesini istiyorsa, sizin kaynak dosyalarınızı `-Wall`, `-Wextra` , `-Werror`, flaglarini kullanarak derleyip çıktı dosyalarını üretecek olan Makefile dosyasını oluşturmanız gerekmektedir. Makefile dosyasını oluştururken cc kullanın ve Makefile dosyanız yeniden ilişkilendirme yapmamalıdır (relink).
- `Makefile` dosyanız en azından `$(NAME)`, `all`, `clean`, `fclean` ve `re` kurallarını içermelidir.
- Projenize bonusu dahil etmek için Makefile dosyanıza `bonus` kuralını dahil etmeniz gerekmektedir. Bonus kuralının dahil edilmesi bu projenin ana kısmında kullanılması yasak olan bazı header dosyaları, kütüphaneler ve fonksiyonların eklenmesini sağlayacaktır. Eğer projede farklı bir tanımlama yapılmamışsa, bonus projeleri `_bonus.{c/h}` dosyaları içerisinde olmalıdır. Ana proje ve bonus proje değerlendirmeleri ayrı ayrı gerçekleştirilmektedir.
- Eğer projeniz kendi yazmış olduğunuz `libft` kütüphanesini kullanmanıza izin veriyorsa, bu kütüphane ve ilişkili `Makefile` dosyasını proje dizinindeki `libft` klasörüne ilişkili `Makefile` dosyası ile kopyalamanız gerekmektedir. Projenizin `Makefile` dosyası öncelikle `libft` kütüphanesini kütüphanenin `Makefile` dosyasını kullanarak derlemeli ardından projeyi derlemelidir.
- Test programları sisteme yüklenmek zorunda değildir ve puanlandırılmayacaktır. Buna rağmen test programları yazmanızı şiddetle önermekteyiz. Test programları

sayesinde kendinizin ve arkadaşlarınız projelerinin çıktılarını kolaylıkla gözlemleyebilirsiniz. Bu test dosyalarından özellikle savunma sürecinde çok faydalanacaksınız. Savunma sürecinde kendi projeleriniz ve arkadaşlarınızın projeleri için test programlarını kullanmakta özgürsünüz.

- Çalışmalarınız atanmış olan git repolarına yüklemeniz gerekmektedir. Sadece git reposu içerisindeki çalışmalar notlandırılacaktır. Eğer Deepthought sizin çalışmanızı değerlendirmek için atanmışsa, bu değerlendirmeyi arkadaşlarınızın sizin projenizi değerlendirmesinden sonra gerçekleştirecektir. Eğer Deepthought değerlendirme sürecinde herhangi bir hata ile karşılaşılırsa değerlendirme durdurulacaktır.

- Çalıştırılacak dosya `pipex` olarak adlandırılmalıdır.

- You have to handle errors sensitively. In no way can your program quit unexpectedly (Segmentation fault, bus error, double free, etc). If you are unsure, handle the errors like the shell command `< file1 cmd1 | cmd2 > file2`.

- Hataları güzel bir şekilde ele almalısınız. Programınız beklenmedik bir şekilde sonlanmamalıdır (Segmentation fault, bus error, double free, etc). Eğer emin değilsen, hataları örnekteki shell komutu gibi ele al `< file1 cmd1 | cmd2 > file2`.

- Programınızda memory leak olmamalıdır.

- Aşağıdaki fonksiyonları kullanmanıza izin verilir:
  - `access`
  - `open`
  - `unlink`
  - `close`
  - `read`
  - `write`
  - `malloc`
  - `waitpid`
  - `wait`
  - `free`
  - `pipe`
  - `dup`
  - `dup2`
  - `execve`
  - `fork`
  - `perror`
  - `strerror`
  - `exit`

Bölüm III

Hedefler

Amacınız pipex programını kodlamaktır.
Bu şekilde uygulanmalıdır:

```
$> ./pipex file1 cmd1 cmd2 file2
```

Her ihtimale karşı: file1 ve file2 dosya adlarıdır, cmd1 ve cmd2 , parametreleriyle birlikte shell komutlarıdır.

Pipex programının çalıştırılması, bir sonraki shell komutuyla aynı şeyi yapmalıdır :

```
$> < file1 cmd1 | cmd2 > file2
```

III.1 Örnekler

```
$> ./pipex infile ``ls -l'' ``wc -l'' outfile
```

Örnekteki gibi olmalıdır “< infile ls -l | wc -l > outfile”

```
$> ./pipex infile ``grep a1'' ``wc -w'' outfile
```

Örnekteki gibi olmalıdır “< infile grep a1 | wc -w > outfile”

Bölüm IV

Bonus bölüm

Bonuslar, yalnızca zorunlu bölümünüz MÜKEMMEL ise değerlendirilecektir.MÜKEMMEL ile kastettiğimiz ,doğal olarak eksiksiz olması gerektiğini ,yanlış kullanımlar vb. gibi kötü hatalar durumunda bile başarısız olmayacağıdır.Bu, zorunlu bölümünüzün notlandırma sırasında TÜM puanları almaması durumunda bonuslarınızın ÖNEMSENMEYECEĞİ anlamına gelir..

- Birden fazla pipe ele almak :

```
$> ./pipex file1 cmd1 cmd2 cmd3 ... cmdn file2
```

Eşdeğer olmalıdır :

```
< file1 cmd1 | cmd2 | cmd3 ... | cmdn > file2
```

- İlk parametre "here_doc" olduğunda « and » ile destekle

```
$> ./pipex here_doc LIMITER cmd cmd1 file
```

Eşdeğer olmalıdır:

```
cmd << LIMITER | cmd1 >> file
```

Bölüm V

Teslim Etme ve Değerlendirilme

Çalışmanızı her zamanki gibi `GiT` repository' nize gönderin . Yalnızca repository'nizdeki çalışma notlandırılacaktır.