İŞLETİM SİSTEMLERİ PROJE ÖDEVİ

2018 GÜZ

Veriliş tarihi: 8.11.2018 Pazartesi

Teslim tarihi: 9.12.2018 Pazar, Saat: 23:59

Teslim yeri: SABİS üzerinden

Notlar:

1) Ödevle ilgili sorunlar için Arş.Grv. Deniz Balta, email: ddural@sakarya.edu.tr ile irtibata geçiniz.

2) Proje C programlama dilinde ve Linux ortamında geliştirilecektir.

TANIM

Bu ödevin amacı, proses (süreç) yönetimi, I/O (Giriş/Çıkış) ve Linux signal kullanımının temellerini öğrenmektir. Projede sizin göreviniz temel bir **Kabuk** uygulaması geliştirmektir. Kabuk bir komut satırı yorumlayıcısıdır: Kullanıcıdan alınacak komutlar için sürekli olarak standart girişi okur, karşılık gelen programları (varsa) çalıştırır ve kullanıcı tarafından sonlanana kadar çıktılarını görüntüler. Bu projede, ana bir prosesten (kabuk prosesi) yeni prosesler oluşturabilmeli, komutları yürütebilmeli ve I/O yönlendirmesi (redirection) yapabilmelisiniz.

Linux Kabuk (Shell)

Genel olarak kabuk için komut satırının amacı, komut istemi (prompt) görüntülemek ve sonra kullanıcı tarafından girilen metnin okunmasıdır. Girilen metin, tam olarak tek bir komut veya bir dizi komut içeriyor olabilir. Komut, kabuğun kendi ortamı içinde tanımlanan bir komutun adıdır (builtin komut) ya da bir yürütülebilir dosyanın tam yolu (adı ile birlikte) ve ardından isteğe bağlı argümanlarıdır. Yürütülebilir dosya adı ve bağımsız değişkenler karakter adedi/türleri ile sınırlandırılmıştır. Bu nedenle çalıştırılabilir adlar ve bağımsız değişkenler boşluk içermeyen dizelerdir.

Yönlendirme:

Kabuk, her komutun standart çıktı ve girdisini yeniden yönlendirebilir. Örneğin **"> echo 12 > test.txt**" girerseniz, echo işleminin çıktısı, yönlendirme nedeniyle test.txt dosyasına yazılır. Ardından, **"> increment < test.txt**" komutunu girdiğinizde ise test.txt dosyasındaki girdi okunarak artımdan sonra ekrana 13 yazdırılır. (Not: Burada "increment" adlı standart girişten aldığı tamsayıyı bir arttıran yürütülebilir dosyanın bilgisayarda, bilinen bir dizinde olduğu varsayılmıştır.)

```
> echo 12 > test.txt
> increment < test.txt
13
```

Boru (pipe):

Her komutun arasında sınırlayıcılar kullanılarak, standart bir kabuk üzerinde sıralı komutlar yürütmek de mümkündür. Bu projede **boru** (|) ve **noktalı virgül** (;) sınırlayıcılarına odaklanacağız. **Boru** durumunda, her komutun standart çıkışı, ondan sonra gelen komutun standart girişine bağlanır. Örneğin, aşağıdaki komut, komutun ekosunun çıktısının daha önce kurulan bağlantıyı kullanarak artış girişine transfer edildiği 13'ü basmalıdır. Artış, bu durumda basit bir artım sürecidir.

```
> echo 12 | increment
13
```

Noktalı virgül durumunda ise süreçler arasında bir bağlantı yoktur. Komutlar soldan sağa doğru yürütülür. Örneğin, aşağıdaki komut ilk önce 12 yazdırmalı, 2 saniye uyuyacak ve sonra 13 yazmalıdır.

```
> echo 12; sleep 2; echo 13
12
13
```

Sıralı komutlar noktalı virgül kullanılarak yürütüldüğünde, her komut için I/O yönlendirmesi ve borulama (pipe) yapılması gerektiğini unutmayın.

Arkaplan işleme, komutların tamamlanmalarını beklemeden ve hemen her **arkaplan** komutundan sonraki komutun eşzamanlı yürütülmesidir. **Arkaplan** yürütme, komutun sonuna bir ve işareti **(&)** ekleyerek gerçekleştirilir.

İSTENENLER:

Puanlama:

Yapılacak İşler	**Puan**
Prompt	**5**
Built in komut	**5**
Tekli komut icrası	**10**
Giriş yönlendirme	**15**
Çıkış yönlendirme	**15**
Arkaplan çalışma	**20**
Boru (pipe)	**20**
Dokümantasyon, düzen	**10**
TOPLAM	**100**

1) **Prompt (5 Puan):**

Başladıktan ve her komutun tamamlanmasından sonra veya her arka plan komutunun hemen ardından, kabuk ">" basmalıdır. Komut istemini yazdırdıktan sonra, programınızın doğru derecelendirilmesi için standart çıktıyı çoğaltmak temeldir.

Print buffer'ı boşaltmak için aşağıdaki veya eşdeğer bir kod kullanılmalıdır:

```
printf("> ");
fflush(stdout);
```

2) **Quit (5 Puan) (Built-in komut)**

Kabuk, quit komutu ile sonlandırılmalıdır.

```
> quit
```

Tek istisna, önceki komutun arka planda hala çalıştığı zamandır. Bu durumda, programınız yeni komutlara yanıt vermeyi durdurmalı, tüm arka plan işlemlerinin bitmesini bekleyecek, ilişkili bilgileri yazdıktan sonra hemen sonlandırmalıdır.

3) **Tekli komutlar (10 Puan):**

Kabuk bağımsız değişkenlerle veya bağımsız değişkenlerle tek bir komutu okuduğunda komutu çalıştırmalı, komut tamamlanıncaya kadar engellemeli ve ardından istemine geri dönmelidir. Kabuk bir alt proses oluşturmalı ve onu komutla belirtilen programa dönüştürmelidir. Örnek:

```
> ls -l
```

Yukarıdaki örnekte, programınız kendisini bir ls prosesi haline getiren bir alt süreç oluşturmalıdır. Programınız daha sonra ls işleminin tamamlanmasını beklemeli ve daha sonra komut prompt'unu görüntüleyerek yeni bir komut beklemelidir.

İlgili LINUX system çağrıları: fork, exec, wait ve/veya waitpid

4) **Giriş yönlendirme (20 Puan):**

Girişe "komut < GirişDosyası" şeklinde yeniden yönlendirme (redirection) yapan bir komut girildiğinde, program verilen komutu çalıştırmadan önce alt prosesin standart girdisini verilen giriş dosyasına yönlendirmelidir. Hiçbir girdi dosyası yoksa, "**Giriş dosyası bulunamadı"** yazdırılmalıdır.

Örnek:

> cat < file.txt
> file.txt'nin içinde ne varsa ekrana yazdırılır.
> cat < nofile.txt
> nofile.txt giriş dosyası bulunamadı.

İlgili LINUX system çağrıları: dup2

5) **Çıkış yönlendirme (20 Puan):**

Çıktısı "komut > çıkışDosyası" şeklinde yeniden yönlendirme (redirection) yapan bir komut girildiğinde, program alt prosesin standart çıktısını çıkışDosyası'na yönlendirmelidir.

Örnek:

> cat file1 > file2

İlgili LINUX system çağrıları: dup2

6) **Arkaplan çalışma (20 Puan):**

Bir komut '&' ile bittiğinde arka planda çalıştırılmalıdır. Komutun tamamlanması için kabuk engellenmemelidir ve hemen komut prompt'u görüntülenmelidir. Daha sonra kullanıcı yeni komutlar girilebilmelidir. Bununla birlikte, arka plan işlemi sona erdiğinde, kabuk çıkış durumunu proses kimliği ile birlikte, aşağıdaki bilgileri kullanıcıya bildirmelidir.

[pid] retval: <exitcode>

Örnek:

> sleep 5 &
> cat file.txt
> file.txt'nin içindekiler ekrana yazdırılır.
> [24617] retval: 0

Burada, ikinci prompt kabuk tarafından derhal yazdırılır, oysa pid ve dönüş değeri arka plan işleminin sonlandırılmasından sonra yazdırılır.

Arka planda çalışan bir proses varken, kabuk bir **quit** komutu alırsa; prompt'u görüntülemesi ve yeni komut girişi durdurmalı, sonra tüm arka plan işlemleri sona erene kadar beklenmelidir. Daha sonra ilgili bilgileri bastırmalı ve derhal çıkmalıdır.

İlgili LINUX system çağrıları: sigaction, WEXITSTATUS

7) **Boru (pipe): (20 Puan):**

Birden fazla komut birbiriyle borular aracılığıyla bağlandığında, i. komutunun çıkışı (i + 1) inci komutunun girişine beslenir. Son komutun çıkışı, dosyaya yönlendirilmediği sürece standart çıktıya bağlanır. İlk komutun girişi, işlemin giriş gerektirmesi durumunda standart girdiden başka bir dosyadan da yönlendirilebilir.

Örnek:

> find /etc | grep ssh | grep conf

Bu görevde, komut promptu görüntülenmeden önce, pipe içindeki tüm komutların sonlanması gerektir.

İlgili LINUX system çağrıları: pipe, dup2. Ayrıca proseslerin stdin ve stdout dosya descriptörlerine de bir göz atın.

Geri döndürülecekler:

1) Kaynak dosyalar: *.c ve *.h dosyaları
2) Makefile
3) Readme dosyası

Tüm kaynak dosyalarınızda, Makefile ve Readme dosyasında grup elemanlarınızın adları yazılı olmalıdır. Dosyaları sıkıştırıp "grpXogrYsubZ.rar" şeklinde göndermenizi bekliyoruz. Burada X: grup nonuzu, Y: öğretim nonuzu (gündüz için 1, gece için 2) ve Z: şube harfini ifade ediyor.

Tekrar hatırlatalım, proje C programlama dilinde ve Linux ortamında geliştirilecektir.

Kolay gelsin…