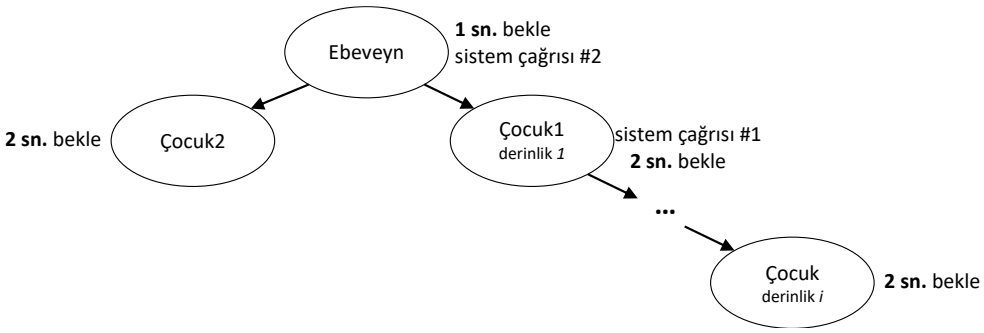
Hafta 6, Laboratuvar 3, Uygulama 2: Proses Yaratma ve Çalıştırma

Çalışmanızı *UZEM*'deki yükleme alanına yükleyiniz!
Uygulama prosedürünü terminal kullanarak gerçekleştiriniz.

Prosedür: Aşağıdaki proses ağacını oluşturan kodu yazınız.

- Proses ağacı, kullanıcının argüman olarak girdiği sayısal değere göre derinleşmektedir (*Hint*: `$ ./uyg2 3`). Kullanıcının girdiği değerin değişiminden bağımsız **Ebeveyn** prosesin iki adet çocuğu vardır. Ağacın sadece **Çocuk1**'in olduğu tarafı derinlik bilgisine göre oluşturulur.
- Kullanıcı, programı çalıştırırken terminal *option*(lar)ı girebilir (*Hint*: `getopt() <---> #include <unistd.h>`). *Örnek*: `$ ./uyg2 -o 4`
  `-i` : Öğrenci numarası ve isim bilgisi programın en üst satırında bastırılır. *Örnek*: `$ ./uyg2 -i 0`
  Çıktı: `152120151028 @sergenasik wrote this.`
- Doğru derinlik kurgusu için derinlik minimum değeri **2** olarak varsayılacaktır. Terminal argümanları kontrolü bu doğrultuda gerçekleştirilmelidir. *Hint*: `int dummyint = atoi(char); //char* to integer in c`
- Her bir proses kendisinin ve ebeveyninin proses ID'sini ekrana yazdırmalıdır.

- Kaynak kod içerisinde **minimum sayıda fork()** sistem çağrısı satırı yer almalıdır.
  - Geçersiz örnek #1:
    ```
    myFunc(){fork();}
    ```
  - Geçersiz örnek #2:
    ```
    goto line x ...
    line x: fork();
    ```



- Her bir proses, şekilde gösterilen ve alt adımlarda istenen bekleme haricindeki ek işlemler için maksimum bir tane sistem fonksiyonu çağırmalıdır. Sistem çağrısı sonuçları üst satırında bilgilendirme cümlesi (*Hint*: `$ echo`) olacak şekilde (*Hint*: *çıktı yönlendirme*) proses temelli *log* dosyasında tutulmalıdır (**logAll.log**).
  - **Ebeveyn** proses ilgili komutu içeren sistem fonksiyonu ile dizine **logs** isminde bir klasör ve içerisine *log* (**logAll.log**) dosyasını oluşturarak tarih bilgisini basar (*Hint*: `$ date`). Sonrasında diğer proseslerin de oluşturulması için **1** saniye bekler (Bu beklemenin amacı semboliktir ve diğer proseslerin belirtilen hiyerarşiye göre düzenli bir şekilde oluşmasını sağlamaktır). Ardından, ilgili komutu gerçekleştiren sistem fonksiyonu ile o anda işletim sistemi üzerinde koşan uygulamalar arasından yalnızca üstteki proses ağacını PID bilgileriyle birlikte **logs** klasörü altındaki *log* dosyasına basar ve işlemini tamamlar.
    Hint: `$ man pstree $ pstree <options?> <PID>`
    Hint: `char mychar[size]; sprintf(mychar, "%s %d", dummytext, dummyint); //hybrid string+int to char* in c`
  - **Çocuk1** (derinlik1), *log* dosyasının içeriğini ekrana bastırır (önceki çalıştırmaların sonuçları dosya içerisinden görülüyor olacak, mevcut çalıştırma ayrıca birazdan basılacak).
  - **Ebeveyn** haricindeki diğer tüm prosesler derinlik bilgisine göre kendisinden sonraki prosesi yarattıktan sonra **2** saniye bekler (bu beklemelerin amacı da semboliktir ve proses ağacının düzenli bir şekilde görülmesini sağlamaktır) ve işlemini tamamlar.

Not: Yukarıda belirtilen her bir işlemin sıralamada ayrıca bir önemi bulunabilir. Doğru sonuç için istenenleri dokümandaki sıralama ile gerçekleştiriniz.

Yüklenmesi Gereken Dosyalar: **<OgrNo>** için öğrenci numaranızın son altı hanesini giriniz!

kaynak kod dosyası	**<OgrNo>_uyg<#><Şube>.<dil>**
*çalıştırma sonuçlarını içeren çıktı dosyası	**<OgrNo>_uyg<#><Şube>_output.txt**
log dosyası	**<OgrNo>_uyg<#><Şube>_logAll.log**

*derlenmiş bir *executable* dosya değildir, koşu sonuçlarını içerir!

Puanlama Sistemi:

	QUIZ	PERFORMANS	
doğru ağaç	**40**	**15**	hakimiyet
argüman(lar)	**20**	**10**	kod düzeni
sistem çağrıları		**15**	

Uygulamalar laboratuvar performansı ve uygulamanın doğru kısımlarına göre değerlendirilir. Yüklenmesi gereken dosyalar "**<OgrNo>_<Ders><Şube><AkademikYıl><GUZ/BHR/YAZ>_<UYG/HW><#>.zip**" (**Örnek: 152120151028_IsSisLabC2223BHR_UYG2.zip**) isimlendirme formatında sıkıştırılarak yükleme alanına yüklenir. Yükleme hatalarına ceza puanı uygulanır. Sisteme yüklenmeyen çalışmalar geçersiz sayılır.