**به نام خدا**


**دانشگاه صنعتی امیرکبیر**
**( پلی تکنیک تهران )**

سیستم‌های عامل (بهار ۱۴۰۱)

# پاسخنامه تمرین دوم

**استاد درس:**

دکتر جوادی

1) با توجه به فراخوان سیستمی fork مشخص کنید و توضیح دهید که خروجی هر بخش شامل چه عبارتی است؟

الف)

```
int main()
{
    fork() && fork() && fork() && fork();
    printf("+");
    return 0;
}
//
Each child returns 0 in fork and each parent returns 1
so we only have the parents in each fork and it will give us 5 +
in the end
+++++
```

ب)

```
int main()
{
    fork() || fork() && fork() || fork() && fork() || fork();
    printf("+");
    return 0;
}
// 12 times -> just a shape of processes and theirs childs is enough
```

2) زمانی که فراخوان سیستمی fork صدا زده می‌شود معمولا یکی از دو فرآیند (والد یا فرزند) یک فراخوان سیستمی دیگر به نام exec را صدا می‌زنند. تحقیق کنید و توضیح دهید که این فراخوان سیستمی چه کاری انجام می‌دهد و چرا یکی از دو فرآیند (والد یا فرزند) پس از اجرای دستور fork آن را صدا می‌زنند؟

فراخوانی سیستمی )exec معمولاً پس از دستور )fork در یکی از پرداز ههای پدر یا فرزند اجرا میشود تا فضای حافظهٔ پردازه را با برنامهٔ دیگری جایگزین کند. در حالت عادی اگر پس از فراخوانی سیستمی )fork، ()exec را صدا نزنیم پردازهٔ فرزند ادامهٔ کد پردازه پدر را اجرا خواهد کرد. در حقیقت پدر و فرزند برنامهٔ یکسانی را اجراخواهند کرد، در صورتی که اغلب هدف از ایجاد پردازهٔ جدید، انجام کاری مجزا از پردازهٔ اصلی است. برای مثال فرض کنید برنامهٔ سنگینی داریم که نیاز به خواندن دادههایی از روی فایل دارد. میخواهیم این خواندن از فایلها را توسط پرداز های مجزا انجام دهیم تا در صورت به وجود آمدن مشکلی در این کار، در روند برنامهٔ اصلی خللی ایجاد نشود. در صورتی که بخواهیم عمل خواندن از فایل را در کد خود پردازهٔ اصلی بنویسیم زمان فراخوانی )fork تمام فضای حافظهٔ این برنامه باید کپی شود که حجم زیادی از RAM ما را اشغال خواهد کرد. بنابراین برنامهای جدید و مجزا برای این کار مینویسیم و پس از فراخوانی )fork از طریق دستور )exec برنامهٔ جدایی که نوشتهایم را بر روی یکی از پرداز ها اجرا میکنیم (و طبیعتاً دیگری به اجرای برنامهٔ اصلی ادامه خواهد داد). از این طریق هم از مزیت استفاده از دو پردازه بهره میبریم و هم نیاز به کپی کردن تمام فضای حافظهٔ برنامهٔ اصلی نداریم. به علاوه به دلیل این که کدهای مربوط به هر بخش را از هم جدا کردهایم از پیچیده و نامفهوم شدن کد مربوط به پردازهٔ اصلی جلوگیری خواهد شد.

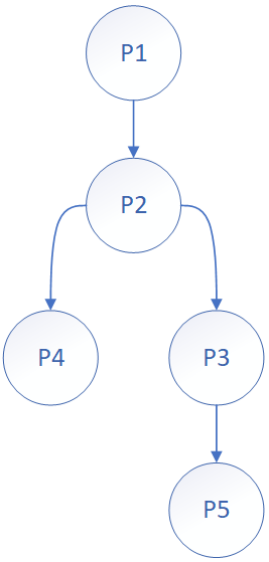
3) لیست تغییرات وضعیت پردازه (process state) را از بدو اجرای برنامه زیر تا اتمام آن را بیان کنید. مشخص کنید هر تغییر وضعیت قبل یا بعد کدام خط از کد اتفاق می‌افتد. فرض کنید پردازه‌ی زیر تنها پردازه‌ی موجود در سیستم است.

```
1.int i = 1;
2.while (i < 100) i++;
3.printf("%d", i);
4.while (i > 0) i--;
5.printf("%d", i);
```

1. ابتدا پردازه ساخته شده و وضعیت آن new است.
2. بعد از بارگذاری کد در حافظه و آماده شدن برنامه وضعیت آن به حالت ready تغییر می‌کند.
3. بعد از اختصاص پردازه به پردازنده وضعیت آن به حالت running در می‌آید و خطوط ۱ و ۲ اجرا می‌شوند.
4. خط ۳ یک دستور I/O است پس وضعیت پردازه به waiting تغییر پیدا می‌کند و بعد از اتمام اجرای آن به حالت ready در می‌آید.
5. در زمان اجرای خط ۴ حالت پردازه به running تغییر وضعیت می‌دهد.

6. در خط آخر مجددا دستور مورد نظر، یک دستور I/O است پس وضعیت پردازه به صورت waiting درآمده و بعد از اتمام آن به صورت ready در می‌آید.
7. در انتها نیز وضعیت برنامه به terminated تغییر می‌کند.

4) شبه‌کدی بنویسید که درخت فرآیند زیر را بوجود آورد، بگونه‌ای که:




- P5 برنامه‌ای بنام sort را اجرا می‌کند.
- P4 برنامه‌ای بنام search را اجرا می‌کند.
- P2 تا خاتمه‌ی فرآیند P3 صبر می‌کند و سپس فرآیند P4 را از بین می‌برد.

```
// P1
int pid3, pid4;
if (fork() == 0) { //P2
   if ((pid3 = fork()) == 0) // P3
      if (fork() == 0) // P5
         exec("sort");
   else if ((pid4 = fork()) == 0) // P4
      exec("search");
   else {
      waitpid(pid3);
      waitpid(pid4);
```

}

}

برنامه‌ی فوق یک نمونه از جواب صحیح است و شبه کدی که عملکرد مشابه را داشته باشد صحیح است.

5) با توجه به فراخوانی سیستمی fork به سوالات زیر پاسخ دهید

الف) در برنامه ی زیر چند بار ۲ چاپ می شود؟مراحل را بنویسید.

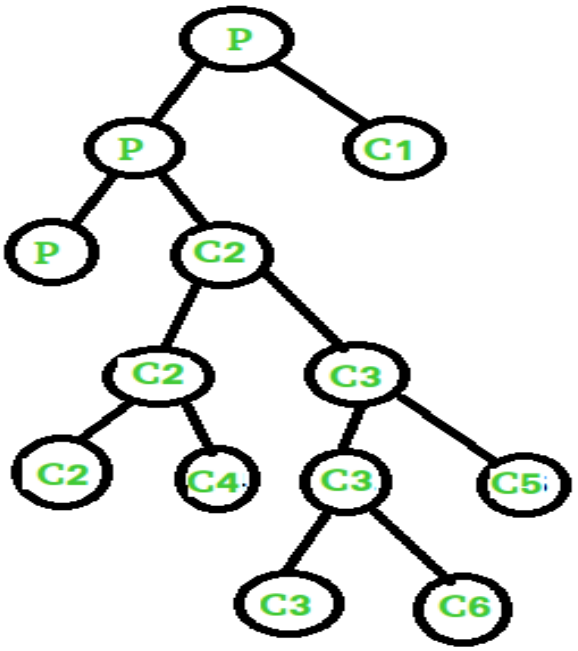
```
int main()
{
    if(fork() && (!fork())){
        if(fork() || fork())
            fork();
    }
    printf("2");
    return 0;
}
```

در این سوال 7بار 2 چاپ میشود. به صورت زیر:

1. Fork دو فرآیند ایجاد می کند که یکی P والد (pid برابر شناسه ی فرزند جدید) و دیگری پردازه ی فرزند C1 (شناسه فرآیند = 0) است.
2. 2. در عبارت if از عملگر AND استفاده می کنیم (یعنی &&) و در این مورد اگر شرط اول نادرست باشد، شرط دوم را ارزیابی نمی کند و 2 را چاپ نمی کند. P شرط دوم را بررسی میکند و دو پردازه جدید ایجاد می کند (یکی P و دیگری C2). در شرط دوم از عملگر NOT استفاده می کنیم که برای پردازه ی فرزند C2 مقدار true را برمی گرداند و دستور if داخلی را اجرا می کند
3. Child C2 دوباره دو پردازه جدید ایجاد می کند (یکی C2 به عنوان والد و C3 به عنوان فرزند) و از عملگر OR (یعنی ||) استفاده شده که اگر شرط اول نادرست باشد شرط دوم را بررسی میکند.

   C2 (والد) قسمت if را اجرا میکند و دو پردازه ی جدید ایجاد می کند. (یکی C2 (والد) و یکی C4(فرزند)).

   C3(فرزند) شرط دوم را چک میکند و دو پردازه جدید میسازد.(یکی C3 (والد) ویکی C5 (فرزند) )
4. C3(والد) وارد قسمت if میشود دو پردازه جدید دیگر ایجاد میکند. ( C3(والد) و C6 (فرزند))


P
P
C1
P
C2
C2
C3
C2
C4
C3
C5
C3
C6

ب) در برنامه ی زیر مشخص کنید چند بار hello پرینت میشود. مراحل را بنویسید.

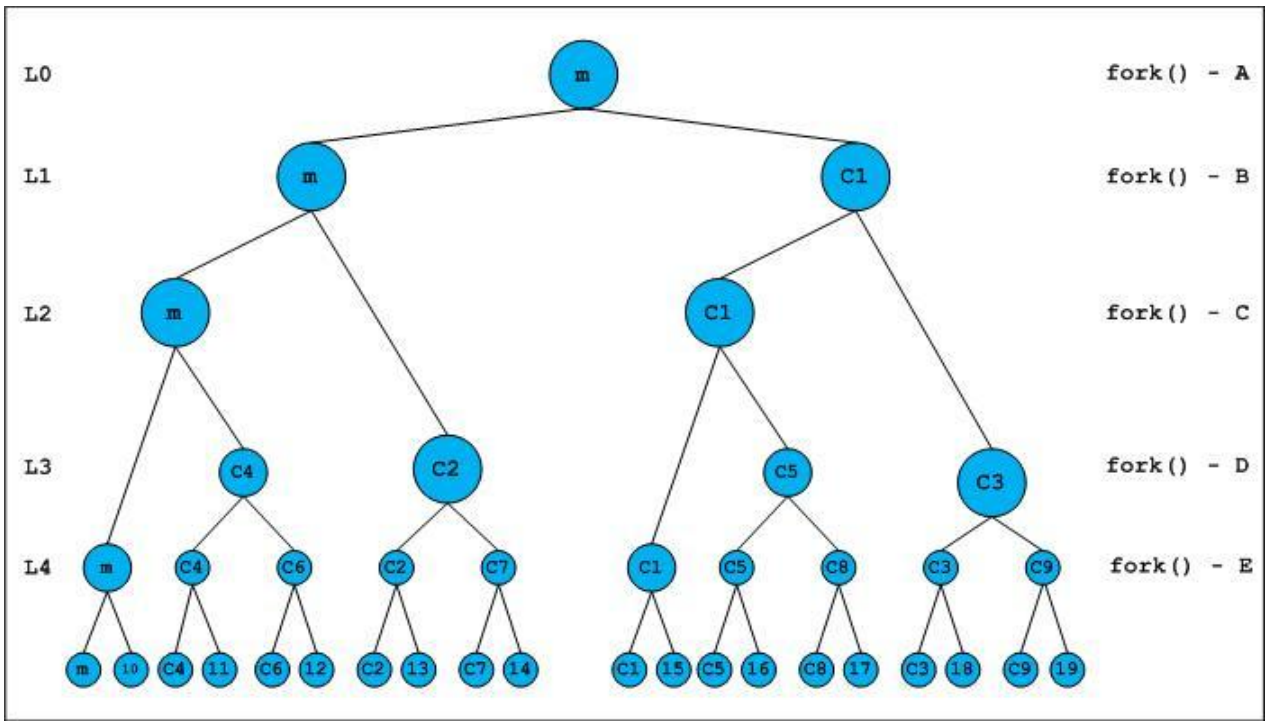
```
3  int main()
4  {
5      fork();
6      fork() && fork() || fork();
7      fork();
8
9      printf("hello\n");
10     return 0;
11 }
```

پاسخ:

```
#include <stdio.h>
int main()
{
   fork(); /* A */
   ( fork()  /* B */ && fork()  /* C */ ) ||    fork(); /* D */
   fork(); /* E */

   printf("hello\n");
   return 0;
}
```

در این برنامه 20 بار خروجی چاپ میشود. که یکی پردازه ی main است و 19 تای دیگر پردازه هایی هستند که با فورک توسط main ایجاد شده است .




موفق باشید

تیم تدریس‌یاری درس سیستم‌های عامل