تمرین دوم درس سیستم عامل

سوال اول

در این سوال parent و child هر کدام متغیر های a=30 خود را دارند زیرا این متغیر global نیست، یا به عبارتی share نشده است پس برای فرزند (pid==0) مقدار تغییر یافته a برابر 40 خواهد شد و برای والد مقدار تغییر یافته a برابر 20 خواهد شد. آدرس های خروجی یکسان هستند اما میدانیم آدرس ها در scope هر process هستند زیرا virtual address اند پس یکسان بودن آدرس ها ربطی به share شدن متغیر ندارد چون هر متغیر در یک context جدا تعریف شده. (دو متغیر به یکجا اشاره ندارند!)

در نتیجه خروجی:

20 0x7ffdef72d7e8

40 0x7ffdef72d7e8

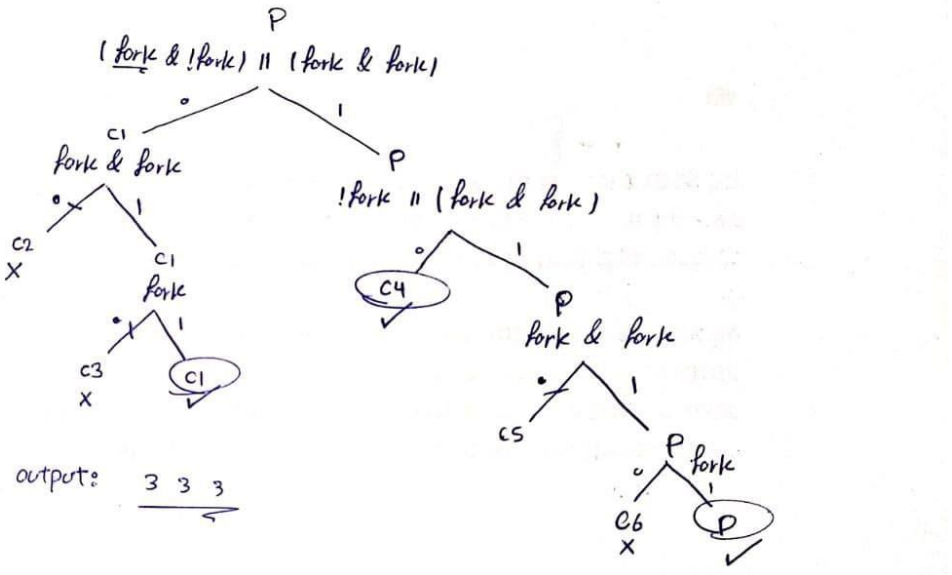
(Or – Due to processes running in parallel)

40 0x7ffdef72d7e8

20 0x7ffdef72d7e8

سوال دوم

طبق عکس زیر، هر باید مرحله به مرحله بر حسب اینکه خروجی ()fork چه شود حالت بندی کنیم. توجه کنید در هنگام && اگر اولین شرط false باشد شرط دوم چک نمیشود (اجرا هم نمیشود طبیعتا) و در || اگر اولین شرط true باشد شرط دوم چک نمیشود (به طبع اجرا هم نمیشود).



تمرین دوم درس سیستم عامل

سوال سوم

یک zombie process، زمانی ایجاد میشود که process بچه وظایف خود را انجام داده و بعد از اتمام تمام محتویات خود را clear میکند اما برای free شدن خود، process والد این کار را باید انجام دهد. از زمانی که process بچه کار خود را اتمام کند تا زمانی که والد آن را حذف کند وارد حالت zombie میشود.

زمانی که والد یک process فرزند terminate شود (به هر دلیلی، میتواند تمام شده باشد یا از خارج بسته شده باشد) و دیگر منتظر فرزند نماند، به فرزند orphan میگوییم.

از آنجایی که یک zombie process فقط زمانی نابود میشود که parent دستور دهد. وظیفه OS آن است که مطمئن شود بعد از اتمام parent تمام فرزندانش هم نابود شوند. (یکی از کار های init اعمال wait برای فرزند توسط والد است) پس دستور wait مانع از بوجود آمدن orphan processes شده و zombie processes ها به طور خودکار نابود میشوند. این وظیفه Kernel است که از این اتفاق اطمینان حاصل کند.

سوال چهارم

بازه زمانی	پردازه و فعالیت در حال اجرا + زمان هر فعالیت	Queue ()
0 - 150	P1 (150)	**(P1)** - P2 - P3 - P4
150 – 400	P2(100) + IO (400 – Ignored due to DMA) + P2(50)	**(P2)** – P3 – P4 – P1
400 - 550	P3 (150)	**(P3)** – P4 – P1 – P2
550 – 700	P4(50) + MM (50) + P4(50)	**(P4)** – P1 – P2 – P3
700 - 850	P1 (50) + fork() + P1(100)	**(P1)** – P2 – P3 – P4 – P11
850 – 1000	P2(100) + IO(400 – Ignored du to DMA) + P2(50)	**(P2)** – P3 – P4 – P11 – P1

جدول بالا به این صورت است که در هر سطر بازه زمانی سپری شده برای هر پردازنده، فعالیت های دقیق انجام شده و صف Queue است که آیتم داخل پرانتز و پررنگ شده در حال اجرا است (در حال اجراست و هنوز تمام نشده).

میدانیم ()fork فرزند خود را به ته صف برده و IO کلاک از CPU استفاده نمیکند به دلیل DMA اما MM برای فعالیت به کلاک پردازنده نیاز دارد پس حساب میشود

سوال پنجم

در کل 20 بار عبارت hello چاپ میشود. کافی است برنامه را دو بخش کنیم. بخش اول مربوط به شرط اول و بخش دوم مربوط شرط دوم است. در مرحله بعد کافی است تمام حالات بخش اول را بنویسیم که میشود 4 تا. حال یک branch از بخش دوم را نوشته که 5 خروجی دارد، و چون برای هر خروجی بخش اول همه خروجی های بخش دوم تکرار میشود کافیست 4x5=20 کنیم و تعداد حالات کلی بدست می آید.

تمرین دوم درس سیستم عامل