به نام خداوند رنگین کمان

امیرحامد گودرزی کالجی　　۹۹۳۱۰۹۹　　تمرین سری ۴ درس سیستم عامل

(۱)

الف) درباره به اشتراک گذاری منابع میان فرایند والد و فرزند ۳ گزینه محتمل داریم که با توجه به قصد طراح دخ دادن هرکدام، ممکن است:

- والد و فرزند تمام منابع را به اشتراک بگذارند
- فرزند زیر مجموعه‌ای از منابع والد را به اشتراک بگیرد
- والد و فرزند هیچ منبعی به اشتراک نگذارند (سیستم عامل ویندوز از این مدل بهره می برد)

ب) abort شدن یک فرایند توسط والدش، می تواند به دلایل زیر باشد:

- فرزند استفاده غیرمجاز از منابع ببرد
- وظیفه داده شده به فرزند انجام شده یا دیگر نیاز نباشد.
- فرایند والد در حال تمام شدن باشد و سیستم عامل به فرایند فرزند اجازه ادامه دادن، ندهد

ج) با نگاه اول به کد، می توان دید که ساختار صحیحی دارد و فرایند والد و فرزند را به درستی جدا و کدنویسی کرده است اما در هست فرایند والد از دو تابع استفاده کرده‌ایم که کتابخانه‌شان فراخوانی نشده است:

wait(NULL) ← باید کتابخانه <sys/wait.h> را به کد اضافه کنیم.

exit(0) ← با یکتابخانه <stdlib.h> را به کد اضافه کنیم

با این حال نیز، حتی اگر کتابخانه‌ها را شامل نکنیم نیز، کد کامپایل شده و به درستی کار می‌کند، کد والد نیز اجرا می‌شود. خروجی کد به شکل زیر است:

code　code.c

child　complete

• TAS یک مکانیزم برای synchronization سازی است که به صورت سخت افزاری اجرا می شود و به صورت atomic می باشد. مقادیر آن به صورت true, false یا ۱ و ۰ می تواند باشد. مقداری که به آن می دهیم متغیری برای lock است برای همین ۰ و ۱ به معنی غیرقفل یا قفل بودن می باشد.

کارکردش نیز به شکلی است که مقدار یک lock را می گیرد. آن را در یک متغیر دیگر ذخیره می کند، مقدار اصلی (آدرس) lock را true می کند، مقدار قبلی اش را که در یک متغیر ذخیره کرده است برمی گرداند.

استفاده از آن در فرایندها نیز به گونه ای است که تا زمانی که مقداری که به ما می دهد فرایند کاری نمی کند و در یک حلقه خالی سرگردان است، هنگامی که lock را برداشته شد و false شد، فرایند به کارش ادامه می دهد. برای اینکه زمانی که خواستیم به حافظه بنویسیم مناسب است.

• Prevention: هنگامی که فرایندی برای منابع مورد نیازش درخواست می دهد، از قبل حساب کنیم که آیا محتمل است ددلاک به وجود آید یا نه اگر محتمل بود اجازه رخ دادن فرایند را ندهیم.

avoid: به این گونه است که ددلاک به وجود می آید اما به دست آن نمی رویم، حسابی کنیم که آیا ددلاک ما را وارد حالت غیرایمن می کند یا نه، اگر کرد به دستش نمی رویم.

detection & recovery: تشخیص دادن ددلاک و حل کردن آن

deadlock ignorance: با این روش، روش شترمرغی نیز می گویند که در این روش ددلاک را نادیده می گیریم و مانند یک شترمرغ سرمان را در شن می کنیم :)

• در مسئله ۵ فیلسوف حال اگر فیلسوفی نتواند دوتا شاخ مجاورش را برای خوردن بردارد، هرچه که برداشته است را باید روی میز گذاشته و به فکر کردن بپردازد و بعدا دوباره تلاش کند. این موضوع به شرط hold & wait در شرایط ددلاک ها اشاره دارد. یعنی اگر هیچ فیلسوفی نتواند تا شاخ دومی باشد که همه باید منتظر بمانند و باعث ایجاد deadlock می شود. حال می توان به اضافه کردن تا شش شاخ، شکل دومین تا شش هر فیلسوف را حل کند. البته مسئله جدیدی که ایجاد می شود این است که تا شش شاخ به کدام فیلسوف برسد و موجب ایجاد livelock می گردد و فیلسوفان دچار قحطی (به معنای واقعی!) می شوند و به هیچ کدام غذایی نرسد.

(۳) یک صف به سایز N برای N فرایند درنظر می‌گیریم. برنامه‌ای بنویسید که به نحوی که هر فرایند خودش را در صف اضافه کند به شکلی که در یک زمانی از صف بیرون بیاید و مجاز باشد که ناحیه بحرانی خود را اجرا کرده و از صف خارج شده با این روش می‌توان تضمین کرد که هر فرایندی زمانی می‌تواند ناحیه بحرانی خود را اجرا کند که درون صف باشد.

حال دو آرایه turn و flag به سایز N ایجاد می‌کنیم به شکل زیر:

flag [Pid] ⇐ سایز N دارد و Pid بین ۰ تا N-1 می‌باشد

turn[i] = Pid ⇐ سایز N دارد.

دو تابع lock(), unlock() تعریف می‌کنیم. lock() بدین صورت است که فرایند صبر می‌کند تا زمانی که به پایان صف برسد.

unlock نیز بدین صورت است که خروج از صف را شناسه گذاری می‌کند.

اجرای ناحیه بحرانی زمانی اتفاق می‌افتد که فرایند در آخر صف باشد که همان بین lock() و unlock() است.

نحوه تعریف توابع می‌تواند به شکل زیر باشد:

```
lock (Pid) {
  for i to N {
      turn[i] = Pid;
      flag[Pid] = i;
      while ( for all k != Pid and flag[k] < i) or (turn[i] != Pid))
      { ; }
  }

unlock (Pid) {
      if ( Process exists or terminates)
          flag [Pid] = -1;
      else
          flag [Pid] = 0
}
```

(۱۴) کد داده شده در این سوال، مربوط به برنامه ای است که یک فرایند فرزند میسازد و فرایند فرزند نیز در خود یک ترد تولید می کند که متغیر global variable ، value را برابر با پنج قرار میدهد. اما خواسته سوال از ما، مقدار value در فرایند والدی باشد.

از آنجایی که در هنگام تولید فرایند فرزند تمامی متغیرها کپی و در جای دیگری از حافظه اشغال می شوند، می توان گفت value برای فرایند فرزند متفاوت از value برای فرایند والد است.

پس تردی که در فرایند فرزند مقدار value را تغییر می دهد، تاثیری بر مقدار value در فرایند والد نخواهد گذاشت.

! تقریبا به توضیحات بالا، خروجی برنامه در خط P به صورت زیر می باشد:

PARENT: value = 0