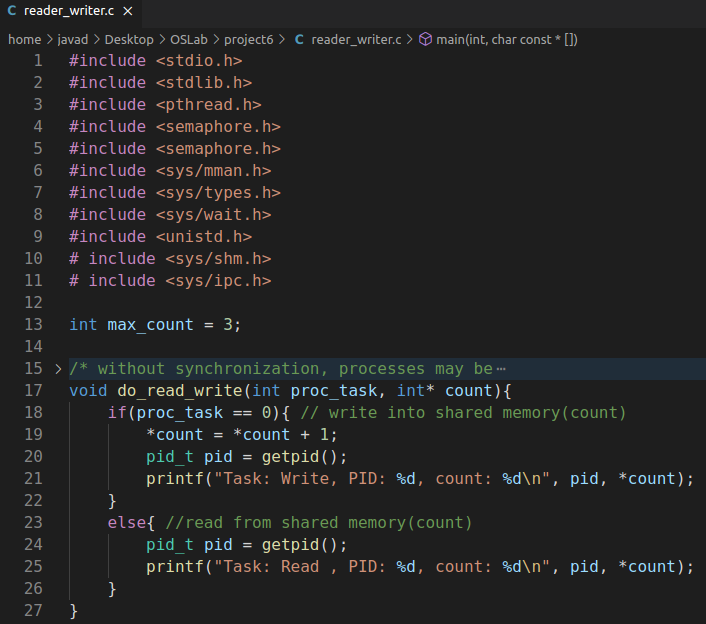
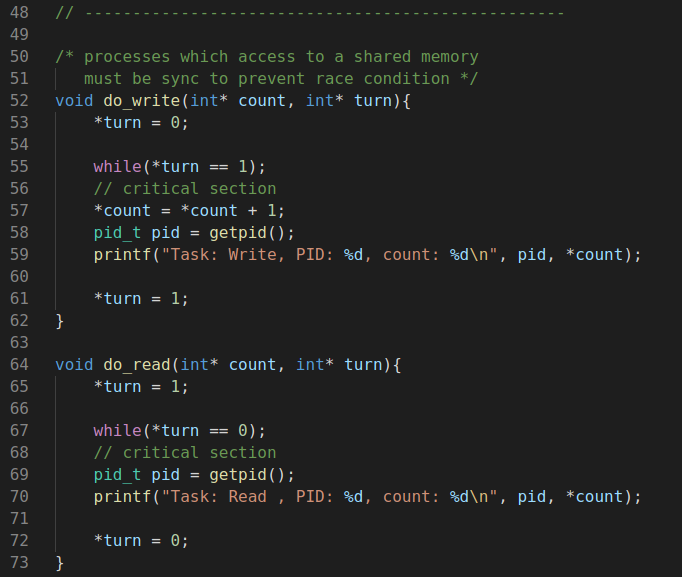
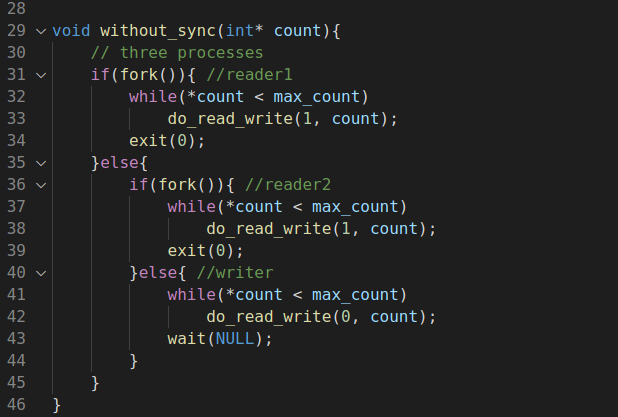
به نام خدا

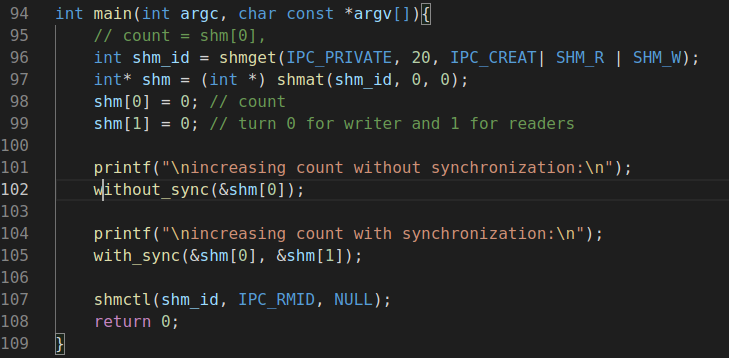
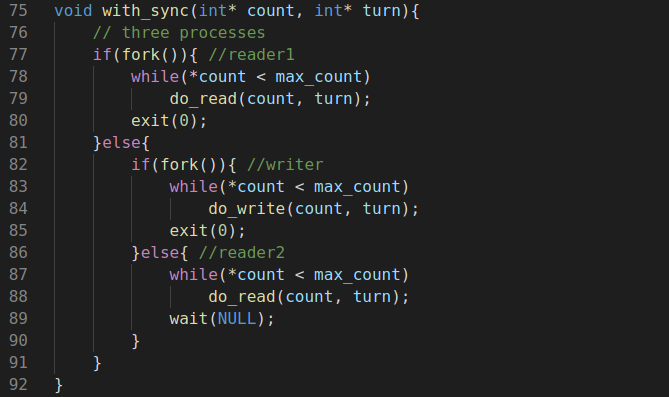
محمد جواد زندیه 9831032

آزمایش شماره 6 سیستم عامل

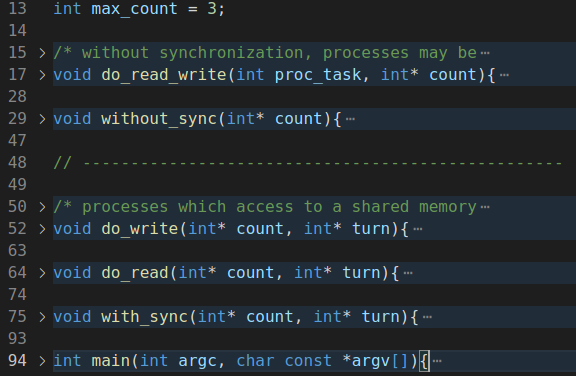
بخش اول: مساله خوانندگان-نویسندگان را پیاده سازی کنید.





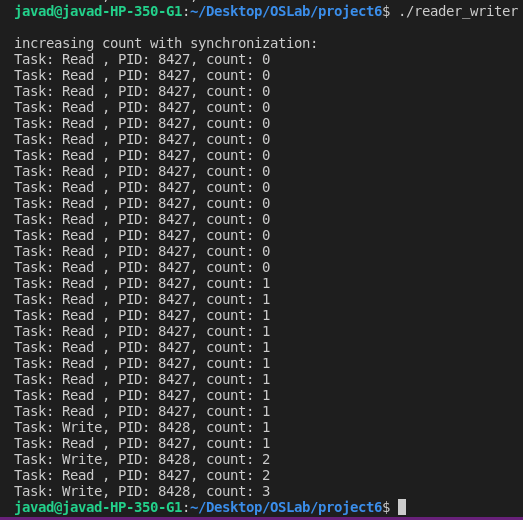


یک شمای کلی از کد به این صورت می باشد



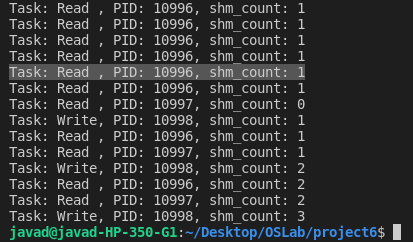
در این مسئله دو عدد reader و یک عدد writer داریم که reader ها همزمان اجازه دسترسی به ناحیه بحرانی یعنی خواندن مقدار count دارند اما reader و writer همزمان اجازه دسترسی به مقدار count را ندارند و باید با گذاشتن قفل روی ناحیه بحرانی از دسترسی همزمان خواننده و نویسنده به مقدار count جلوگیری کنیم. در این مسئله چون reader ها همزمان می توانند به مقدار count دسترسی داشته باشند در واقع میتوان مسئله را به مسئله Peterson تبدیل کرد و برای خواننده ها با turn = 1 و برای نویسنده با turn = 0 ناحیه بحرانی را کنترل کرد. این الگوریتم به این صورت عمل می کند که تا نوبت پردازه نرسیده باشد که وارد بشود حق ورود به ناحیه بحرانی را ندارد.

خروجی به این صورت می باشد:

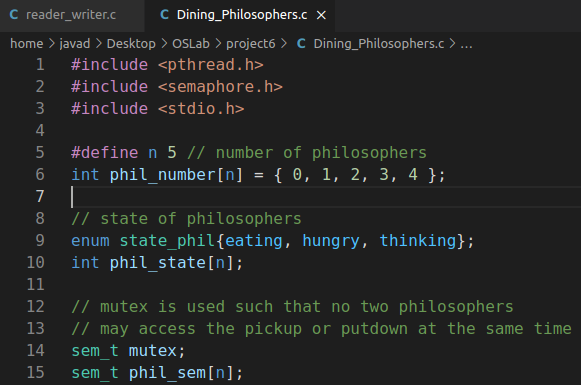


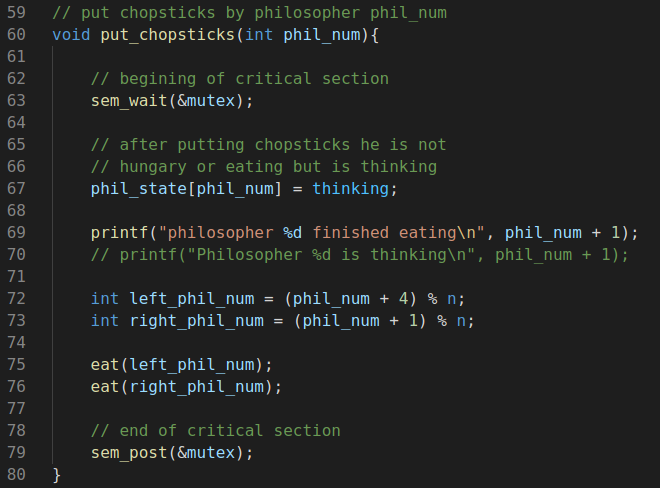
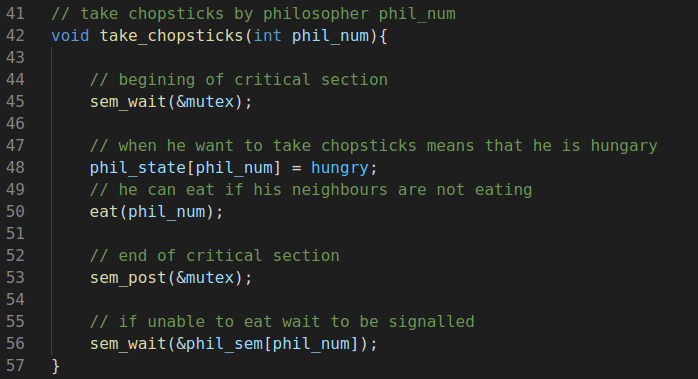
همانطور که مشاهده می شود پردازه ها طوری عمل نکرده اند که وقتی عمل خواندن انجام می شود به عنوان مثال بتواند همزمان عمل نوشتن انجام شود در واقع از روی اینکه مقدار count به صورت صعودی است می توان فهمید که عملیات خواندن به صورت اتومیک صورت گرفته و عملیات نوشتن هم به همین صورت.

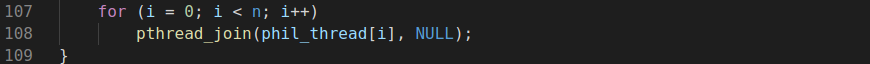
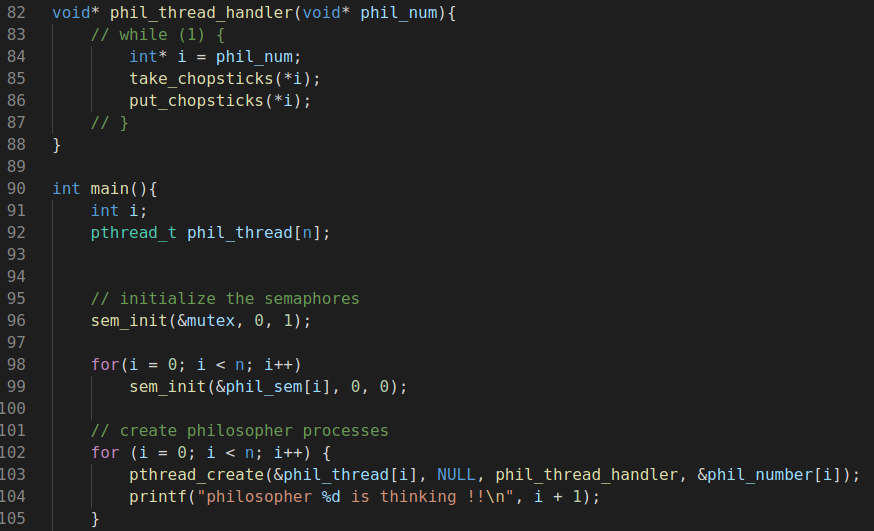
اگر روی ناحیه بحرانی قفل گذاشته نمیشد به این صورت می بود که از به هم ریخته بودن و غیر صعودی بودن مقدار count که چاپ شده است میتوان متوجه شد که پردازه ها بدون توجه به اینکه دسترسی به count بحرانی است همزمان نوشتن و خواندن روی این داده را انجام داده اند.



بخش دوم: مساله فیلسوف های غذا خور

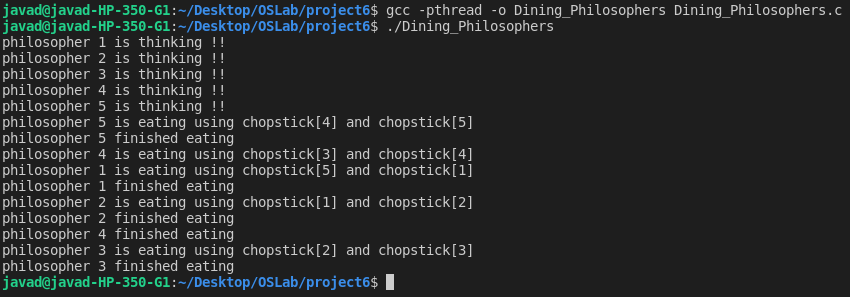






در این مسئله 5 فیلسوف در دور یک میز هستند و قرار است که با 5 چوبی که روی میز است غذای وسط میز را بخورند. هر فیلسوف در کنار خود 2 فیلسوف چپ و راست را دارد و همچنین یک چوب چپ و یک چوب راست هم وجود دارد و برای آنکه بتواند بخورد باید هر دو چوب کنار خود را در اختیار داشته باشد. در این مسئله از ترد یا ریسمان برای هر یک از فیلسوف ها استفاده شده است و یک عدد سمافور برای غذای وسط میز و 5 عدد هم سمافور برای هر یک از فیلسوف ها در نطر گرفته شده است. هر فیلسوف می تواند در یکی از سه حالت hungry , thinking, eating باشد. در هندلر مربوط به هر یک از ترد ها 2 کار انجام میشود. ابتدا چوب ها را بر میدارد ( که این عملیات به دلیل اینکه چوب ها با همسایگان مشترک است نیاز به سمافور دارد که بتواند هر دو چوب کناری خود را در صورتی که آزاد هستند در اختیار بگیرد) و سپس شروع به خوردن می کند (که نیاز است که دو همسایه او در حال خوردن نباشند و چوب ها آزاد باشند) و سپس بعد از اینکه خوردن غذا این فیلسوف تمام شود چوب ها را روی میز میگذارد و همچنین به دو همسایه خود نیز اطلاع می دهد که چوب ها را گذاشته و آنها می توانند استفاده کنند (تعارف میزنه بهشون😊) و این عملیات می تواند تا تعداد بیشمار یا تعداد معینی ادامه یابد.

خروجی:



همگی فیلسوف ها زمانی که سر میز می آیند ابتدا در حالت thinking هستند. سپس یکی از آنها شروع به خوردن می کند و وقتی کارش پایان یافت اطلاع finished را به همسایه ها می دهد و ...

در این جا با این شرط کار انجام شده است که هر یک از افراد فقط یک مرتبه غذا بخورند اما میتوان بخش while در تابع phil\_thread\_handler را فعال کرد تا به صورت بی نهایت کار ادامه پیدا کند.