دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

نام و نام خانوادگی: محمدحسین ستاک شماره دانشجویی: ۹۶۳۱۸۱۱

گزارش پروژه پایانترم سیستم عامل

سوال ۱)۱,۱)

در این سوال قفل ticketlock با استفاده از struct spinlock و شبیه به آن درست شده است. و این به این معنی است که ایت قفل دارای busy waiting می باشد. در ساختار ticketlock دو متغیر int به نام های curticket و lastticket مسئول کنترل بلیت ها هستند. Curticket تیکتی را نشان می دهد که فرایند با آن تیکت در حال گرفتن سرویس می باشد. Lastticket تیکت آخری که به فرایندی داده شده را نگه می دارد. هر فرایند منتظر می ماند تا curticket با ticket خود برابر شود و سپس سرویس می گیرد.

همانطور که گفته شد در صورت سوال فراخوان های سیستمی ticketlockinit و ticketlockTest نیز اضافه شده اند. و در user program ، ticketTest این دو سیستم کال تست شده اند.

نتیجه:

```
$ ticketTest
child adding to shared counter
got ticket pid : 5      currticket: 0   lasTicket: 1
child adding to shared counter
got ticket pid : 6      currticket: 1   lasTicket: 2
child adding to shared counter
got ticket pid : 7      currticket: 2   lasTicket: 3
child adding to child adding to shared counter
got ticket pid : 9      currticket: 3   lasTicket: 4
shared counter
got ticket pid : 8      currticket: 4   lasTicket: 5
child adding to shared counter
got ticket pid : 10     currticket: 5   lasTicket: 6
child adding to shared counter
got ticket pid : 11     currticket: 6   lasTicket: 7
child adding to shared counter
got ticket pid : 12     currticket: 7   lasTicket: 8
child adding to shared counter
got ticket pid : 13     currticket: 8   lasTicket: 9
child adding to shared counter
got ticket pid : 14     currticket: 9   lasTicket: 10
user program finished
got ticket pid : 4      currticket: 10  lasTicket: 11
ticket value : 10
$
```

نام و نام خانوادگی: محمدحسین ستاک شماره دانشجویی: ۹۶۳۱۸۱۱

گزارش پروژه پایانترم سیستم عامل

(۲,۱

با استفاده از قفل ticketlock، مسئله خوانندگان-نویسندگان شبیه سازی می شود. User program مانند نمونه داده شده در صورت سوال با نام rwTest می باشد. متغیر rwVariableTest حکم داده مشترک دارد. نویسنده ها سعی بر اضافه کردن به آن دارند. خواننده ها نیز می خواهند این مقدار را بخوانند. خواننده ها میتوانند با هم و تا زمانی که خواننده ای دیگر در حال خواندن است، شروع به خواندن کنند. اما نویسنده ها باید یکی یکی و بدون حضور کس دیگری متغیر را تغییر دهند.

با مشخص کردن pattern ترتیب ورود نویسنده ها و خواننده ها مشخص می شود.

خروجی با pattern برابر ۱۹ یعنی: 10011

```
init: starting sh
$ rwTest
enter pattern for readers/writers test
10011
child adding to shared counter
reader from shared counter : 0
child adding to shared counter
reader from shared counter : 0
child adding to shared counter
writer from shared counter
child adding to shared counter
writer from shared counter
user program finished
last value of shared counter: 2
$
```

دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

نام و نام خانوادگی: محمدحسین ستاک شماره دانشجویی: ۹۶۳۱۸۱۱

گزارش پروژه پایانترم سیستم عامل

سوال ۲) ۱,۲)

در این بخش مقدمات thread سازی سیستم عامل انجام می شود. این thread ها ترد های سطح کرنل می باشند. تفاوت این ترد با ترد های سطح کاربر: ترد های کاربر توسط کاربر و در سطح کاربر، ایجاد می شوند. یعنی سیستم عامل این ترد ها را به عنوان ترد نمی شناسد. اما ترد کرنل بر عکس توسط خود سیستم عامل پیاده سازی می شوند. پیاده سازی آن پیچیده تر، نیازمند پشتیبانی سخت افزاری (atomic بودن) و برای تعویض متن به زمان بیشتری نیازمندند. از طرفی هنگامی که یک ترد سطح کاربر block می شود، دیگر ترد ها به کار خود ادامه می دهند، اما در ترد کرنل تمامی ترد ها ار بین رفته و کل فرایند بلاک می شود.

برای پیاده سازی ساختار جدیدی به نام thread ایجاد شد. فیلد های لازم متنقل (مانند kstack و state و context و tf و chan) و فیلد های لازم اضافه شد.

هر فرایند با استفاده از ساختار threads ترد های خود را نگه می دارد. که آرایه ای از ترد ها و همچنین قفلی برای این آرایه می باشد.

برای همگام سازی با این ساختار تغییرات متعددی نیز در فایل های دیگر صورت گرفت. اگر به لیست فرایند ها نگاه کنیم می بینیم دو فرایند init و sh، فرایند دارای ۱۶ (MAX_THREAD = 16) ترد می باشد که ترد اول آنها sleep می باشد و بقیه unused می باشند.

```
init: starting sh
$ 1 1 sleep  init 80104017 80104154 80104bbe 80105bd0 801059bf1 0 unused init1 0 unused ini
t1 0 unused init1 0 unused init1 0 unused init1 0 unused init1 0 unused init1 0 unused init
1 0 unused init1 0 unused init1 0 unused init1 0 unused init1 0 unused init1 0 unused init1
 0 unused init
2 2 sleep  sh 80103fdc 801002ca 80100fac 80104ec2 80104bbe 80105bd0 801059bf2 0 unused sh2
0 unused sh2 0 unused sh2 0 unused sh2 0 unused sh2 0 unused sh2 0 unused sh2 0 unused sh2
0 unused sh2 0 unused sh2 0 unused sh2 0 unused sh2 0 unused sh2 0 unused sh2 0 unused sh

$
```

۲,۲)

در این قسمت سیستم کال های createThread و exitThread و joinThread و getThread پیاده سازی شد. تابع createThread مشابه fork، exitThread مشابه exit، joinThread مشابه wait می باشد که شبیه آنها پیاده سازی شد.

در user program testThreadSystemCalls ترد جدیدی درست می کنیم و تابعی را برای انجام به آن می دهیم.