Bu derste mutex kullanımı ve güvenlik ve performans arasındaki denge üzerine yoğunlaşarak zamanlama hatalarını detaylarıyla gözden geçireceğiz.

Ssnserver

Ders kişiler/yaşları/sosyal güvenlik numaralarının veritabanını oluşturan bir kod üzerine. Ana kod ssnserver.c. bu, bir kişinin ismi üzerine şifrelenmiş (sırasıyla soyismi, ismi) bir kırmızı-siyah ağaç oluşturur. Val alanı bir Entry alanına işaret eder ve bu alanda tekrardan dizi halinde, kişinin adı, yaşı, ve sosyal güvenlik numarasını barındır.

Ssnserver.c ağacı yaratır ve Standard giriş bilgilerinden 4 çeşit giriş bilgisi kabul eder.

1. **ADD *fn ln age ssn*** – bu, ağaca entry ekler.
2. **DELETE *fn ln*** – bu, ağaçtan entry siler.
3. **PRINT** – bu ağacı print eder.
4. **DONE** – bu programın çıkış yapmasını sağlar.

Bir deneyelim:

```
UNIX> ssnserver
ADD Jim Plank 31 987-65-4321
PRINT

_____________________________________________
Plank, Jim                    -- 987-65-4321   31
---------------------------------------------
ADD Phil Fulmer 45 123-45-6789
ADD Pat Summitt 42 111-11-1111
PRINT

_____________________________________________
Fulmer, Phil                  -- 123-45-6789   45
Plank, Jim                    -- 987-65-4321   31
Summitt, Pat                  -- 111-11-1111   42
---------------------------------------------
DELETE Jim Plank
DELETE Steve Spurrier
Error: No Steve Spurrier
PRINT

_____________________________________________
Fulmer, Phil                  -- 123-45-6789   45
Summitt, Pat                  -- 111-11-1111   42
---------------------------------------------
DONE
UNIX>
```

İnputgen

Okey, şimdi inputgen.c ye bir bakın. Ssnserver i alt etmek için yazdığım bir program. İnput olarak birkaç event, rasgele numara dağılımı ve bir soyadı dosyası. Oluşturduğum soyadları Ins, yani basitçe /usr/dict/words un yerel bir dosyaya kopyalanmış hali. Program soyadları İns dizininine okur ve 65 önisim için fns dizini vardır. Şimdi, bunun işlevi ssnserver.c nevents random input events yaratmaktır. İlk 50 event random ADD eventidir,

ve daha sonra, ya ADD,DELETE ya da PRİNT eventi yaratacaktır(5/5/1 oranında). PRİNT ve DONE eventleriyle biter.

Ağacın entrilerine karşılık gelen DELETE eventi yaratmak için, inputgen rb- ağacı kullanır. Bu ağaç rasgele bir rakam üzerine şifrelenmiştir ve val alanı da daha önce eklediği isimlerden biridir. Bir DELETE eventi yarattığında ağaçtaki ilk ismi kullanır – bu rasgele bir isim olacaktır, ismi ağaçtan siler, sonra bu ismi DELETE eventi için kullanır.

Yani, şimdi bu biraz karışık ama anlayabiliyor olmanız gerek. İnputgen kurulur ki, yarattığı event sayısı ne olursa olsun, yönettiği ağaç 50 element civarında olsun diye. Bunu kendi kendinize kanıtlamak için, şunu deneyin:

```
UNIX> inputgen 5 1 lns
ADD Phil Normal 2 631-85-0230
ADD Peyton Negligible 7 339-29-9216
ADD Dave Relate 90 440-26-1032
ADD Carla Joseph 15 961-73-1275
ADD Jamal Lane 43 837-68-7746
PRINT
DONE
UNIX> inputgen 5 1 lns | ssnserver
_____________________________________________
Joseph, Carla                -- 961-73-1275   15
Lane, Jamal                  -- 837-68-7746   43
Negligible, Peyton           -- 339-29-9216    7
Normal, Phil                 -- 631-85-0230    2
Relate, Dave                 -- 440-26-1032   90
---------------------------------------------
UNIX> inputgen 6000 1 lns | ssnserver | tail -60
Storehouse, Jamal            -- 378-84-0504   54
Tar, Sergei                  -- 922-35-6408   65
Tennis, Jamie                -- 699-90-2234   84
Triplicate, Catharine        -- 264-43-8097   17
Turing, LaShonda             -- 569-75-2160   42
Twx, Blanche                 -- 488-36-4112   19
Vale, Wendy                  -- 375-04-9327   49
Villainous, Elizabeth        -- 816-64-5753   58
Xerxes, Mary                 -- 489-82-7899   58
---------------------------------------------

_____________________________________________
Accuse, Katie                -- 270-74-3607   94
Anaglyph, Sandra             -- 611-70-3455   10
Antarctic, Peyton            -- 118-39-0627   32
Atrium, Bill                 -- 988-21-6157    7
Beau, Sergei                 -- 723-35-7731   98
Bedevil, Andrei              -- 685-03-6172   42
Biddable, Laura              -- 507-90-3170   67
Blather, Kim                 -- 889-98-2973   46
Boathouse, Jay               -- 212-66-7283   59
Centum, Sandra               -- 348-92-5649   91
Cockpit, Miles               -- 712-40-8903   27
Cunning, Bill                -- 059-56-2417    3
Deduct, Sergei               -- 436-37-7921   83
Eat, Wendy                   -- 424-66-1180   86
Extraordinary, Dizzy         -- 502-24-9923   98
Finland, Raynoch             -- 665-67-0773   59
Geographer, Mary             -- 609-84-0078   29
Gravel, Mary                 -- 515-74-6403   66
Horton, Laura                -- 884-01-5338   98
```

```
Hothouse, LaShonda              -- 098-38-0969   32
Impediment, Xavier              -- 859-72-2538   65
Inattention, Bruce              -- 198-48-1849   11
Litigate, Helen                 -- 292-95-7974   67
Macdonald, Jana                 -- 190-74-9144   38
Mcintosh, Bill                  -- 580-14-3161   22
Moot, Dizzy                     -- 085-92-1219   33
Nellie, Bruce                   -- 928-58-5623    0
Pantomimic, Leonard             -- 328-47-3183   71
Party, Jamal                    -- 295-15-7017   88
Peripheral, Emily               -- 357-03-3434   21
Portland, Elena                 -- 315-91-3735   83
Punt, Dizzy                     -- 912-12-8252   20
Pyhrric, Emily                  -- 887-51-0852   52
Salesman, Semeka                -- 087-79-3275   14
Sledge, Heather                 -- 872-39-4327   91
Soapstone, Catharine            -- 363-42-3221   92
Sony, Jane                      -- 960-59-0669   68
Spheric, Xavier                 -- 915-24-7348   28
Storehouse, Jamal               -- 378-84-0504   54
Tar, Sergei                     -- 922-35-6408   65
Tennis, Jamie                   -- 699-90-2234   84
Tradesmen, Cindy                -- 860-63-2050   77
Triplicate, Catharine           -- 264-43-8097   17
Turing, LaShonda                -- 569-75-2160   42
Twx, Blanche                    -- 488-36-4112   19
Vale, Wendy                     -- 375-04-9327   49
Villainous, Elizabeth           -- 816-64-5753   58
Xerxes, Mary                    -- 489-82-7899   58
------------------------------------------------
UNIX>
```

Fark edeceksiniz ki yukardaki ağacın 50 elementi var.

Ssnserver i gerçek bir server a dönüştürme

Şimdi, ssnserver1.c ye bir bakın.

Bu, ssnservr ı gerçek server a dönüştürür. Bir priz görevi görür, sonra accept_connection() bağlantısını tanımlar, ve bağlantıyı sağlamak için bir server_thread() dizini oluşturur.

ağacın global bir değişken olması haricinde; server_thread() dizini aynı ssnserver.c gibi çalışır.

Bunu **telnet** ile deneyin. Örneğin, cetus3a nın bir penceresinde sunu yapıyorum:

```
UNIX> ssnserver1 cetus3a 5000
```

Başka bir tanesinde ise şunu yapıyorum:

```
UNIX> telnet cetus3a 5000
Trying 128.169.94.33...
Connected to cetus3a.cs.utk.edu.
Escape character is '^]'.
ADD Jim Plank 31 123-45-6789
ADD Phil Fulmer 45 987-65-4321
PRINT
```

```
Fulmer, Phil                    -- 987-65-4321   45
Plank, Jim                      -- 123-45-6789   31
-------------------------------------------------
DONE
Connection closed by foreign host.

UNIX>
```

Bu gayet iyi çalışır. İnputgen.c yi duy alıcısı olarak çalışması için modifiye ettim – şifre de inclient.c. anlaşılması kolaydır ve duy verimini okuması ve standarda print etmesi için ikinci bir dizin kullanır. Bunu aynı server da deneyin:

```
UNIX> inclient cetus3a 5000 5 1 lns
_________________________________________________
Joseph, Carla                   -- 961-73-1275   15
Lane, Jamal                     -- 837-68-7746   43
Negligible, Peyton              -- 339-29-9216    7
Normal, Phil                    -- 631-85-0230    2
Relate, Dave                    -- 440-26-1032   90
-------------------------------------------------
UNIX>
```

Şimdi ssnserver2.c ye bakın. Bu da aynı ssnserver1.c gibi çalışır sadece birden fazla bağlantıyı aynı zamanda sağlayabilir bunu da her bir bağlantı başına bir server_thread() dalı yaratarak (forking off) ama sunu bilin ki, **t** ye erişim mutexler tarafından korunmaz. Bu bir problem yaratır çünkü, örneğin, bir dizin ağaca element ekliyorken bir diğeri yakınlardaki bir elementi siliyor olabilir. Eğer ilk dizin elementi ekleyemeden kesintiye uğrarsa rb-tree göstergeleri ikinci dizin silmeye çalışırken olmaları gereken yerde olmayabilirler. Bu bir error la sonuçlanacaktır, muhtemelen sistem çöküşü (core dump) ile.

Bunu göstermek için, kill it.sh. adında bir **shell script** yazdım. Bu, verilen sayıda inclient işlemini verilen ssnserver2 serverında fork off eder.

Bir deneyin:Bir alette ssnserver2 yi başlatın.Örneğin ben aşağıdakini yaptım:

```
UNIX> ssnserver2 cetus3a 5002
```

Sonra,cetus4a da 5 inclientın servera 1000 tane eşzamanlı entry göndermesini sağladım.

```
UNIX> sh kill_it.sh cetus3a 5002 1000 5 > & /dev/null
```

Birkaç saniye içinde ssnserver2 çöktü.Bu her zaman olmaz ama genellikle olacaktır.Sebebi ise **t** ye erişimin korunmuyor olmasıdır.

Mutex ekleme

Şimdi ssnserver3.c. ye bakalım.Bu bir bağlantı yaratırken her bir dizinin birbirine kitlendiği bir mutex ekler.Bu t ye erişimdeki problemi çözer,çünkü aynı anda iki dizin birden t ye erişemez.**cetus3a** üzerinde **kill_it.sh** i deneyin:

```
UNIX> ssnserver3 cetus3a 5003
```

Ve **cetus4a** da:

```
UNIX> sh kill_it.sh cetus3a 5003 1000 5 > & /dev/null
```

Çökme olmadı!

Bu yüzden, bu karşılıklı problemi çözer,ama bu tokmakla kağıdı zımbalamaya benzer.Herhangi bir dizine sahip olarak mutex ömrü boyunca kilitlenir,serverı sıralamış oluruz—iki dizin aynı anda hiçbir şey yapamaz ve bu bir performans problemidir.ssnserver4.c bu problemi oldukça standar bir şekilde çözer.Mutexleri her zaman kilitlemek yerine,dizin sadece mutex ağaca eriştiği zaman kilitlenir.Bu ADD,DELETE ve PRINT kodlarının içerisindedir.

Performansı nasıl geliştirdiğini göstermek için,Ssnserver3 ü cetus3a da çaıştırdım,ve eş zamanlı olarak aşağıdaki **client**ları **cetus1a**, **cetus2a**, **cetus4a** ve **cetus5a da** çalıştırdım .

```
UNIX> time inclient cetus3a 5004 20000 0 lns > & /dev/null
```

Client lar sırasıyla 10,37,81 ve 145 saniye sürdü.Bunun nedeni sırasıyla görevlendirilmiş olmalarıydı.Daha sonra aynı testi ssnserver4 ü kullanarak yaptım ve sırasıyla 71, 71, 79 ve 80 saniye sürdü. Açıkca görülüyor ki,ssnserver4 eş zamanlı bağlantıları sağlamakta daha iyi,her ne kadar ortalama **client** süresi ssnserver3 de(68.25 saniye) ssnserver4 den(78 saniye) daha iyi olsa da.

SSNSERVER5

Ssnserver3 ün ssnserver4 ten daha iyi bir ortalama **client service** süresinin olması sizi şaşırttı mı?Bunun olmaması gerekir.Bunun bir sebebi, ssnserver4 de ortalama ağaç boyutu bütün **client**ler için 200 elementten oluşacak olması.ssnserver3 de ise ortalama ağaç boyutu 125 (50 si ikinci client başlamadan çıkan ilk client için.100 tanesi ikinci client için,150 si üçüncü ve 200 tanesi de dördüncü için).Diğer bir sebep client e ağaç print edilirken server mutexi kilitler.Bu fazla zaman alan bir iş,ve bunun anlamı başka hiçbir client işlemi aynı anda yapılamaz.

Ağaç print edilirken mutexin gerçekten kilitlenmesi gerekiyor mu?Aslında hayır.Yardımcı olması için birkaç şeyi arabelleğe(buffering) alabilirsiniz.Mutex kilitlenirken ağacı print edecek bir bağ oluşturmak yerine.Bu biraz zaman alacaktır ama duya/prize?(socket) bir dizgi(string) yazılan zaman kadar değil.Daha sonra bir mutex oluşturursunuz ve bir dizgi yazarsınız.Bu ssnserver5.c ile yapılır.Şunu da bilin ki ağaç boyutunu global bir değişkende tutuyorum ve bu bana gerektiğinde **malloc()** arabelleğinde yardım ediyor .

```
UNIX> time inclient cetus3a 5004 20000 0 lns > & /dev/null
```

18,59,51 ve 52 saniye zamanda oldu.Bu büyük bir gelişme.