CS360 Lab #3 -- Fakemake · Jian Huang · CS360

Şimdi daha eğlenceli bir ödev yapalım. Amaç, “make(1)”in daha kısıtlı ve hafifletilmiş versiyonunu yazmak olsun.Bu size “stat(2v)” ve sistem kullanımı hakkında uygulama yapma imkanı verecek.

Fakemake’nin açıklaması:

Göreviniz bir fakemake programı yapmak. Make gibi, fakemake’nin işlevi de derlemenin otomatikleşmesine yardım etmek.Make’nin aksine, fakemake kendisini bir defa çalıştırılabilecek şekilde sınırlandırır ve derleme yaparken “gcc” kullandığınızı varsayar.

Fakemake’nin yazılış biçimi şöyledir:

fakemake [açıklama-dosya]

Eğer açıklama-dosya belirtilmemişse, “fmakefile”i açıklama dosyası varsayar.Eğer açıklama dosyası yoksa program bir hata vererek kapanabilir.

Açıklama dosyasının her bir satırı şu 5 şeyden biri olabilir :

- Boş bir satır (yani sadece beyaz boşluk içeren bir satır)
- Programı çalıştırılabilir yapmak için kullanılan C kaynak dosyalarının bir belirtimi
- C dosyaların listesi

  Belirtilen tüm dosyalar “.c “ile bitmelidir.Çoklu dosyalar boşlukla ayrılmış olabilir.Liste boş ve C ile başlayan birden fazla satır olabilir.

- C başlık dosyalarının bir belirtimi herhangi bir kaynak dosyası tarafından eklenebilir.
- H dosyaların listesi

  H listesi, C listesi gibi biçimlendirilir ve H ile başlayan birden fazla satır olabilir.

- Çalıştırılabilir dosya için bir tanımlama:
- `E name`

Sadece bir adet çalıştırılabilir dosya olabilir, sadece bir adet E ile başlayan satır olabilir. Eğer E ile başlayan bir satır yoksa bu hata demektir.

- Derleme bayraklarıyla ilgili bir tanımlama:
- `F flags`

Bu bayraklar gcc çağırıldıktan hemen sonra dahil olacaktır. Daha önceden olduğu gibi bayraklar boşluklarla ayrılır ve birden fazla F satırı olabilir. Bu aynı zamanda boş da olabilir.

- Kütüphaneler ya da ekstra nesne dosyalarını adreslemek için bir tanımlama:
- `L list-of-libraries`

Önceden olduğu gibi çoklu dosyalar boşluklarla ayrılır. Liste boş olabilir, L ile başlayan birden fazla satır olabilir. Liste, çalıştırılabilir dosyayı yapan son gcc komutundan sonra dahil edilir.

Fakemake bütün .c dosyalarını .o dosyaları haline getirir (**gcc –c kullanarak**), daha sonra bütün .o dosyalarını çalıştırılabilir dosya haline getirir. **Make** komutu ile benzer olarak, eğer gerek yoksa dosyaları yeniden derlemez. .c dosyalarının derlenip derlenmeyeceğine karar vermek için aşağıdaki algoritmayı kullanır.

- Eğer .c dosyası ile uyuşan .o dosyası yok ise .c dosyası (-c ile) derlenmeli.
- Eğer .c dosyası ile uyuşan .o dosyası var fakat .c dosyası daha güncel ise .c dosyası (-c ile) derlenmeli.
- Eğer .c dosyası ile uyuşan .o dosyası var ve .h dosyalarından herhangibi biri .o dosyasından daha güncel ise .c dosyası (-c ile) derlenmeli.

Eğer exe dosyası var ise ve .o dosyalarından daha güncel ise ve hiçbir .c dosyası tekrar derlenmediyse fakemake exe dosyasını tekrar oluşturmaz. Bu durumun dışında fakemake tekrar exe dosyası oluşturur. (gcc –o kullanarak)

Eğer .c veya .h dosyası belirtilmiş fakat mevcut değil ise fakemake hata vererek çıkmalıdır. Eğer derlemenin ortasında herhangibi bir hata olursa fakemake derhal çıkmalıdır.

Örnek

Örnek için yeni temiz bir dizine girin ve aşağıdakileri yazın

```
UNIX> cp ~huangj/cs360/labs/lab3/* .
UNIX> ls
f.c             f.h             f2.c            makefile        mysort.fm
f.fm            f1.c            lab.html        mysort.c
UNIX> make
gcc -c -g f.c
gcc -c -g f1.c
gcc -c -g f2.c
gcc -g -o f f.o f1.o f2.o
gcc -c -g -I/home/huangj/cs360/include mysort.c
gcc -g -o mysort mysort.o /home/huangj/cs360/objs/libfdr.a
UNIX> f
This is the procedure F1 -- in f1.c
This is the procedure F2 -- in f2.c
UNIX> mysort < f.c
  f1();
  f2();
main()
{
}
UNIX> make clean
rm -f core *.o f mysort
UNIX> ls
f.c             f.h             f2.c            makefile        mysort.fm
f.fm            f1.c            lab.html        mysort.c
UNIX>
```

Bu dizin iki programın kaynak kodlarını bulundurmaktadır. İlk olan ,f, 3 C dosyasından oluşmaktadır bunlar f.c, f1.c, f2.c ve bir başlık dosyası: f.h. İkincisi ise **Rbtree-1** dersinden mysort.c . Makefile make kullanarak bu programların nasıl yapılacağını tanımlayan verileri içerir. F.fm dosyası f'i oluşturmak için bir fakemake dosyası, aynı şekilde mysort.fm dosyasıda mysort'u oluşturmak için bir fakemake dosyasıdır. **~cs360/lab3/fakemake** de bulunan Fakemake exesinin kullanarak deneyin.

```
UNIX> ~cs360/lab3/fakemake
fakemake: fmakefile No such file or directory
UNIX> ~cs360/lab3/fakemake f.fm
gcc -c -g f.c
gcc -c -g f1.c
gcc -c -g f2.c
gcc -o f -g f.o f1.o f2.o
UNIX> touch f.c
UNIX> ~cs360/lab3/fakemake f.fm
gcc -c -g f.c
gcc -o f -g f.o f1.o f2.o
UNIX> rm f
UNIX> ~cs360/lab3/fakemake f.fm
gcc -o f -g f.o f1.o f2.o
UNIX> touch f.h
UNIX> ~cs360/lab3/fakemake f.fm
gcc -c -g f.c
gcc -c -g f1.c
gcc -c -g f2.c
gcc -o f -g f.o f1.o f2.o
```

```
UNIX> touch f.h
UNIX> touch f.o f1.o
UNIX> ~cs360/lab3/fakemake f.fm
gcc -c -g f2.c
gcc -o f -g f.o f1.o f2.o
UNIX> ~cs360/lab3/fakemake f.fm
f up to date
UNIX> f
This is the procedure F1 -- in f1.c
This is the procedure F2 -- in f2.c
UNIX> ~cs360/lab3/fakemake mysort.fm
gcc -c -g -I/home/huangj/cs360/include mysort.c
gcc -o mysort -g -I/home/huangj/cs360/include mysort.o
/home/huangj/cs360/objs/libfdr.a
UNIX> mysort < f.c
  f1();
  f2();
main()
{
}
UNIX> rm f.h
UNIX> ~cs360/lab3/fakemake f.fm
fmakefile: f.h: No such file or directory
UNIX>
```

Gördüğünüz gibi, fm yukarı ifade edilen şekilde olduğu gibi çalışır. Modulu sadece gerekli olduğunda yeniden derler. Fakemake'in ne yapacağı hakkında şüpheye düştünüzde, **~cs360/lab3/fakemake** 'in ne yaptığına bakın.

Detaylar

Açıkça görülüyor ki, programın yaşını test etmek için stat()'ı kullanmak zorunda kalacaksın. Struct stat'ın st_mtime alanı programın yaşı olarak kullanılabilir**.**

Bir string'i gerçekleştirmek için **system()** prosedürünü kullanırsın. Bu verilen string'i Shell command gibi çalıştırır (**sh**, **csh** değil, önemli değil). Eğer geri dönüş değeri 0 olursa, komut başarılıdır. Eğer geri dönüş değeri başka bir şey olursa, komut hatalıdır.

Strateji

Umarım bu bölümde stratejilerinize pek ihtiyacınız olmaz. Ama yardımda bulunmak için programı nasıl yazdığım buradadır :

- Tanımlama dosyasının adını bulmak için kod yazdım. Makefile yaptım ve kodu test ettim.
- Tanımlama dosyasını okumak için ana döngüyü yazdım (dosyalar kütüphanesini kullanarak).Bu döngü sadece **get_line**'ı çağırıyor ve her satırı stdout'a gösteriyor.
- Boş satırları tespit etmek için kod yazıldı.
- C satırlarını tespit etmek için kod yazıldı. Diğer tüm satırlar yok sayıldı.
- Tüm C dosyalarını bir dlist içinde okumak için bir subroutine yazıldı. Bütün tanım dosyası okunduktan sonra, dlist'in içeriği yazdırıldı.
- H satırlarını tespit etmek için kod yazıldı ve bunları başka bir dlist içinde okumak için aynı subroutine kullanıldı. Bu kod test edildi.
- L satırları tespit edilmek için bir kod yazıldı, aynı subroutine tekrar kullanıldı, dosyalar üçüncü bir dlist içerisine atıldı.
- Çalıştırılabilir bir isimde okumak için kod yazıldı, test edildi.
- Tüm F satırlarını tespit etmek ve okumak için kod yazıldı. C, H ve L için kullanılan subroutine tekrar kullanıldı. Bayraklar başka bir dlist içerisinde okundu.
- Herhangi bir işletilmemiş satırdaki hatayı etiketlemek için kod yazıldı. Eğer hiç E satırı yoksa bu da bir hata olarak etiketlendi.
- Header dosyalarını işletmek için kod yazıldı. Bu kod H dlist'ini dolaşır ve her dosya için stat fonksiyonu çağırır. Eğer dosya yoksa bir hata etiketler. Aksi durumda, st_mtime fonksiyonunun en büyük değerini main() programa döndürür.
- C dosyalarının işletimi için kod yazıldı. Bu kod C dlist'i dolaşır ve her dosya için stat fonksiyonu çağırır. Eğer dosya yoksa bir hata etiketler. Aksi durumda, .o dosyasını arar. Eğer bu dosya bulunamazsa veya bu dosya C dosyalarından ya da header dosyalarının en büyük st_mtime değerinden daha az güncelse, dosyanın yeniden yapılması gerektiği söylenir. Bu kod özellikle test edildi.
- .o dosyalarını yeniden yapılandırmak için kod yazıldı. Bu, "gcc –c" string'ini yaratmak ve bunu yazdırmak anlamına gelir. Bu doğru gözüktüğü zaman programın string üzerinde call() fonksiyonunu çağırması sağlandı.
- Herhangi bir dosya yeniden yapılandırılsa da yapılandırılmasa da, herhangi bir .o dosyasının en büyük st_mtime değerini main() programına döndürecek şekilde C dosyası subroutine'i son kez düzenlenerek bitirildi.
- Çalıştırılabilir bir dosya yapılmasına ihtiyaç olup olmadığı test edecek şekilde bir kod yazıldı.

- Sonunda, çalıştırılabilir dosyanın yapılması için kod yazıldı. İlk başta “gcc –c” string’i oluşturuldu, daha sonra bu string yazdırıldı. Test edildikten sonra, programın string üzerinde call() fonksiyonunu çağırması sağlandı.
- Program üzerinde son rötuşlar yapıldı ve test edildi.