CS360 Lecture Notes -- Fields

Jim Plank
Directory: ~plank/cs360/notes/Fields
Ders notları: http://www.cs.utk.edu/~plank/plank/classes/cs360/360/notes/Fields
Wed Aug 25 11:01:12 EDT 1999

Kütüphane alanları, okuma girişlerinin yapılması getchar(),scanf() or gets() kullanmaktan daha kolay olan rutin paketlerdir.
Burda benim yazdığım Unix standartında olmayan bir kütüphane mevcut fakat bu C derleyicisinde çalışabilir(DOS veya Windows'ta içerir)
Eğer dersten sonra kütüphane alanları oluşturmak isterseniz ,
'http://www.cs.utk.edu/~plank/plank/fields/fields.html.' adresinden kaynak kodları alabilirsiniz.

Bu sınıf içindeki prosedür alanları kullanmak için, '/blugreen/homes/plank/cs360/include' dizinindeki fields.h dosyasını include etmen gerekir. C dosyası içine bütün bu yol ismini içeren yolu yazman yerine
sadece #include "fields.h" yazman ve programı ' gcc -I/blugreen/homes/plank/cs360/include ' ile derlemen yeterli.

Obje dosyalarını bağladığın zaman programı çalıştırabilir hale getirmek için, Libfdr notlarındaki talimatları takip etmen gerekir.
Makefile senin için bu talimatlardaki herşeyi yapar.
' printwords.c ' dosyasını gözden geçirdiğin zaman makefile'in 'fields.h' ı bulmasından ve böylece ' libfdr.a ' ile derleme bağlantıları olduğundan emin ol.

```
#define MAXLEN 1001
#define MAXFIELDS 1000

typedef struct inputstruct {
  char *name;               /* Dosya adı */
  FILE *f;                  /* Dosya tanıtıcı */
  int line;                 /* Satır numarası */
  char text1[MAXLEN];       /* Satır */
  char text2[MAXLEN];       /* Çalışma alanları içerir */
  int NF;                   /* Alan numaraları */
  char *fields[MAXFIELDS];  /* Alanlar için pointerler */
  int file;                 /* '1' dosya için , '0' açmak için */
} *IS;

extern IS new_inputstruct(/* dosya adı --stdin için NULL */);
extern IS pipe_inputstruct(/* komut-- stdin için NULL */);
extern int get_line(/* IS */); /* NF'ye döner, ya da dosya sonunda '-1' yazar .Dosya açılmaz   */
extern void jettison_inputstruct(/* IS */);  /* IS' i serbest bırakır ve dosyayı kapatır */
```

Kütüphane alanları ile dosyayı okumak için uygun dosya ismi ile new_inputstruct() fonksiyonunu çağırırız.
New_inputstruct() argümen olarak stdin için NULL değeri , ve sonuç olarak bir IS döndürür. IS bir 'struct inputstruct' ta bir pointer dır.
Bu senin için new_inputstruct() çağrısındaki malloc() dur.Eğer new_inputstruct() dosya açamazsa , NULL değeri dönecek ve başarısızlık nedeni çıktısını perror() ile çağırabilirsin(eğer bunu öğrenmek istersen, perror() üzerindeki man page i oku).

Senin bir IS'in var,satır bilgisini okumak için onun üzerinde get_line() çağırırsın.Get_line() satırın okunuşunu ifade etmek için IS'in durumunu değiştirir.Özellikle:

---> Satırın içeriğini 'text1' içine koyar.

---> Kelimelere satırı bırakır.'NF' alanı alandaki kelimelerin numarasını içerir.'fields' ın ilk 'NF' yerleri 'NF' kelimelerinin herbiri için dizi noktası(ve bu kelimeler null-terminated(boş sonlandırılmış)tır)
---> 'line' alanı satırın satır numarasını içerir.
---> Get_line() dönüş değeri olarak 'NF' alanını döndürür.7
---> Dosya sonuna ulaştığı zaman '-1' değeri döndürür.

Jettison_inputstruct() 'IS' ile ilişkili dosyayı kapatır ve IS'ı serbest bırakır.Şimdilik pipe_inputstruct() için kaygılanma.

Bu prosedürler giriş dosya işlemleri için çok pratiktir.Örneğin;aşağıdaki program(printword.c içindeki) satır numaraları ile giriş dosyasının her ön kelimesini çıkarır.

```
#include < stdio.h >
#include "fields.h"

main(argc, argv)
int argc;
char **argv;
{
  IS is;
  int i;

  if (argc != 2) {
    fprintf(stderr, "usage: printwords filename\n");
    exit(1);
  }

  is = new_inputstruct(argv[1]);
  if (is == NULL) {
    perror(argv[1]);
    exit(1);
  }

  while(get_line(is) >= 0) {
    for (i = 0; i < is->NF; i++) {
      printf("%d: %s\n", is->line, is->fields[i]);
    }
  }

  jettison_inputstruct(is);
  exit(0);
}
```

bu yüzden, örneğin, eğer dosya aşağıdaki 3 satır için 'rex.in' içerirse

```
June: Hi ... I missed you!
Rex:  Same here!  You're all I could think about!
June: I was?
```

sonra rex üzerinde çıktılar çalışır.Sonuçta çıkış aşağıdaki gibidir:

```
UNIX> printwords rex.in
1: June:
1: Hi
1: ...
1: I
```

```
1: missed
1: you!
2: Rex:
2: Same
2: here!
2: You're
2: all
2: I
2: could
2: think
2: about!
3: June:
3: I
3: was?
UNIX>
```

fields.o hakkında önemli bir not da , sadece new_inputstruct() malloc() fonksiyonunu çağırır.Özetle Get_line() IS yapısının alanını doldurur.---
bellek ayırma yönünden verimli değildir.Bu yüzden, sondan 2. satırdaki ilk kelimeyi yazdırmak istediğini varsayalım.Aşağıdaki program(badword.c) çalışmayacaktı:

```
#include < stdio.h >
#include "fields.h"

main(argc, argv)
int argc;
char **argv;
{
  IS is;
  int i;
  char *penultimate_word;
  char *last_word;

  if (argc != 2) {
    fprintf(stderr, "usage: badword filename\n");
    exit(1);
  }

  is = new_inputstruct(argv[1]);
  if (is == NULL) {
    perror(argv[1]);
    exit(1);
  }

  penultimate_word = NULL;
  last_word = NULL;

  while(get_line(is) >= 0) {
    penultimate_word = last_word;
    if (is->NF > 0) {
      last_word = is->fields[0];
    } else {
      last_word = NULL;
    }
  }

  if (penultimate_word != NULL) printf("%s\n", penultimate_word);
```

```
  jettison_inputstruct(is);
  exit(0);
}
```

Niçin? rex.in üzerinde yürüttüğün zaman olanlara bak.

```
UNIX> badword rex.in
June:
UNIX>
```

'Rex:' yerine 'June:' yazar.Çünkü getline() yeni bellek alanın ayırmaz.penultimate_word ve last_word aynı şey için pointerlamayı sona erdirir.
Bu örneği anladığından emin ol, aksi takdirde kafanız karışabilir. 'goodword.c' içindeki programın doğru versiyonu : (strdup() ve free() çağrılarının hepsini içerdiğinden dolayı bu program etkin bir programdır)
İstersen daha iyisini yapabilirsin.

```
#include < stdio.h >
#include < string.h >
#include "fields.h"

main(argc, argv)
int argc;
char **argv;
{
  IS is;
  int i;
  char *penultimate_word;
  char *last_word;

  if (argc != 2) {
    fprintf(stderr, "usage: badword filename\n");
    exit(1);
  }

  is = new_inputstruct(argv[1]);
  if (is == NULL) {
    perror(argv[1]);
    exit(1);
  }

  penultimate_word = NULL;
  last_word = NULL;

  while(get_line(is) >= 0) {
    if (penultimate_word != NULL) free(penultimate_word);
    penultimate_word = last_word;
    if (is->NF > 0) {
      last_word = strdup(is->fields[0]);
    } else {
      last_word = NULL;
    }
  }

  if (penultimate_word != NULL) printf("%s\n", penultimate_word);
  jettison_inputstruct(is);
  exit(0);  }
```

Field.o 1000 karakterden daha az olan tüm giriş satırlarını kapsar.

tailanyf

Şimdi başka bir örnekte, tailanyf.c komut satırı argumeni olarak 'n' alır ve standart girişin son 'n' satırını yazdırır.Standart giriş okumak için kütüphane alanlarını kullanır.