Sistem Programlama

DR. ÖĞR. ÜYESİ ABDULLAH SEVİN

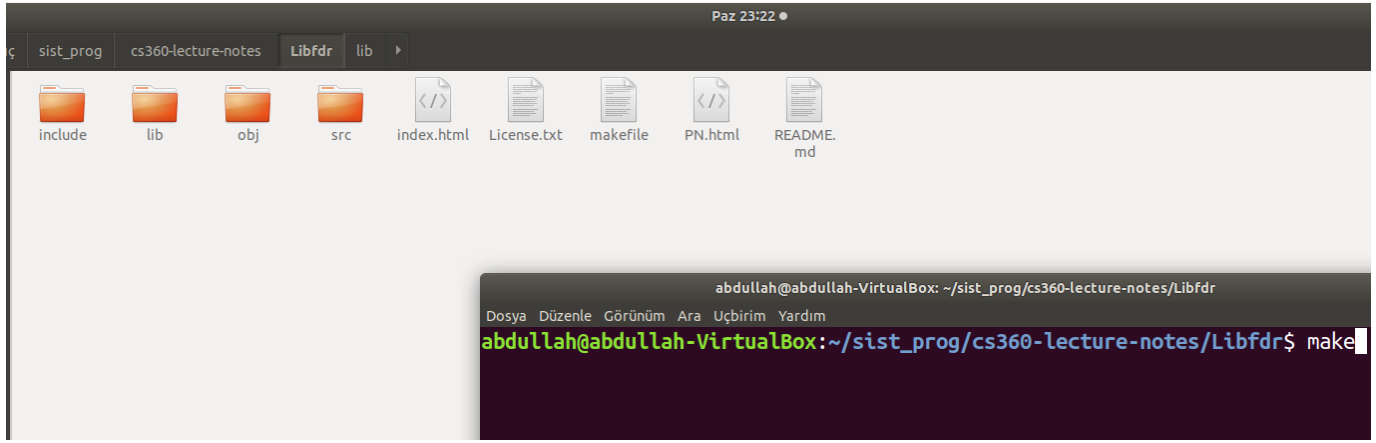
İçerik

a) **Libfdr: Jvals, Fields, Dllists, Red-Black Trees**

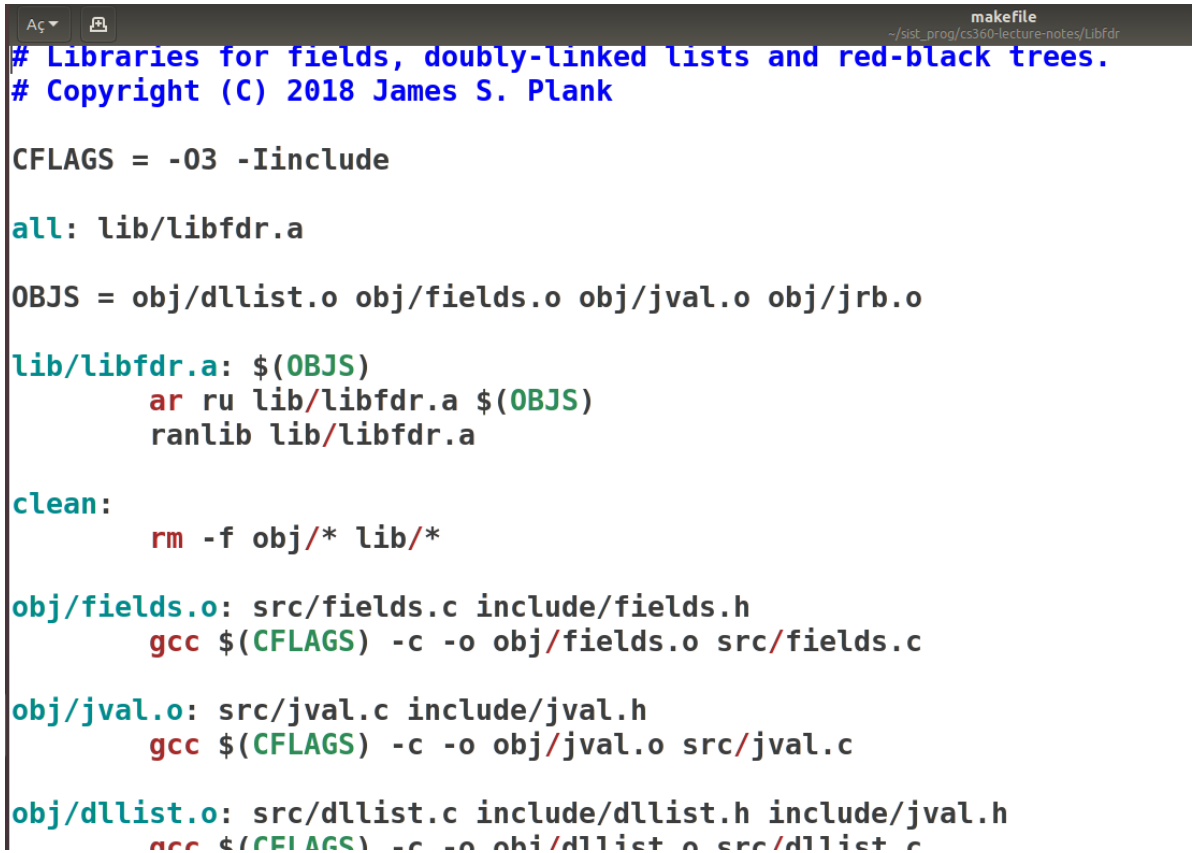
- http://web.eecs.utk.edu/~jplank/plank/classes/cs360/360/notes/Libfdr/index.html
- http://web.eecs.utk.edu/~jplank/plank/classes/cs360/360/notes/Fields/index.html

Jvals, Fields, Dllists, Red-Black Trees

- **Fields** - girdi işlemeyi basitleştiren bir kitaplık
- **Jvals** - genel bir veri türü
- **Dllists** - çift yönlü bağlı listeler için bir kitaplık
- **Red-Black Trees** - kırmızı-siyah ağaçlar için bir kitaplık
- Kütüphanemizi derlemek için dizinde make yapıyoruz



makefile


Aç
makefile
~/sist_prog/cs360-lecture-notes/Libfdr
# Libraries for fields, doubly-linked lists and red-black trees.
# Copyright (C) 2018 James S. Plank
CFLAGS = -O3 -Iinclude
all: lib/libfdr.a
OBJS = obj/dllist.o obj/fields.o obj/jval.o obj/jrb.o
lib/libfdr.a: $(OBJS)
ar ru lib/libfdr.a $(OBJS)
ranlib lib/libfdr.a
clean:
rm -f obj/* lib/*
obj/fields.o: src/fields.c include/fields.h
gcc $(CFLAGS) -c -o obj/fields.o src/fields.c
obj/jval.o: src/jval.c include/jval.h
gcc $(CFLAGS) -c -o obj/jval.o src/jval.c
obj/dllist.o: src/dllist.c include/dllist.h include/jval.h

Fields

- **Fields** kitaplığı, girdi okumayı getchar(), scanf() veya fgets() kullanmaktan daha kolay hale getiren bir rutinler paketidir.
- Kaynaktaki yazarlar tarafından yazılan bir kitaplık -- Unix'te standart değil ama herhangi bir C derleyicisi ile çalışması gerekir (buna DOS/Windows da dahildir).
- Kaynak kodu bu depoda, "Libfdr" dizinindedir.
- Bu sınıftaki **Fields** prosedürlerini kullanmak için, fields.h dosyasını eklemelisiniz.
- Tam yol adını C dosyanıza dahil etmek yerine şunları yapın:

```
#include "fields.h"
```

fields.h

```
/* The fields library -- making input processing easier */

#include <stdio.h>
#define MAXLEN 1001
#define MAXFIELDS 1000

typedef struct inputstruct {
  const char *name;          /* File name */
  FILE *f;                   /* File descriptor */
  int line;                  /* Line number */
  char text1[MAXLEN];        /* The line */
  char text2[MAXLEN];        /* Working -- contains fields */
  int NF;                    /* Number of fields */
  char *fields[MAXFIELDS];   /* Pointers to fields */
  int file;                  /* 1 for file, 0 for popen */
} *IS;

extern IS new_inputstruct(const char *filename);       /* Use NULL for stdin. Returns NULL on failure. */
extern IS pipe_inputstruct(const char *shell_command); /* Returns NULL on failure. */
extern int get_line(IS inputstruct);                   /* returns NF, or -1 on EOF. */
extern void jettison_inputstruct(IS inputstruct);      /* frees the IS and fcloses/pcloses the file */
#endif
```

Fields

- **Fields** kitaplığıyla bir dosyayı okumak için new_inputstruct() öğesini çağırırsınız.
- New_inputstruct(), giriş parametresi olarak dosya adını alır (standart girdi için NULL) ve sonuç olarak bir IS döndürür.
- IS'nin bir struct inputstruct için bir işaretçi olduğuna dikkat edin.
- Bu, new_inputstruct() çağrısında malloc()' tanımlaması mevcuttur.

```
#define talloc(ty, sz) (ty *) malloc (sz * sizeof(ty))
```

- new_inputstruct() dosyayı açamıyorsa, NULL döndürür ve hatanın nedenini yazdırmak için perror()'u arayabilirsiniz (bu konuda bilgi edinmek istiyorsanız, perror()'daki man sayfasını okuyun).

Fields

- IS yapısında, bir satırı okumak için get_line()'ı çağırırsınız. Get_line(), satırın okunmasını yansıtmak için IS'nin durumunu değiştirir. özellikle:

1. Satırın içeriğini text1'e koyar.
2. Satırı kelimelere ayırır. NF field, field'deki kelime sayısını içerir. Fields dizisinin NF slotları, NF sözcüklerinin her birine işaret eder (ve bu sözcükler boş sonlandırılmıştır).
3. line field, satırın satır numarasını içerir.
4. Get_line(), dönüş değeri olarak NF field döndürür.
5. Dosyanın sonuna geldiğinde -1 döndürür.

- Jettison_inputstruct(), IS ile ilişkili dosyayı kapatır ve IS'yi serbest bırakır (serbest bırakır).

src/printwords.c

```c
/* Use the fields library to print each word on standard input, labeled with its line number. */

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

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

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

  /* Open the file as an inputstruct.  Error check. */

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

  /* Read each line with get_line().  Print out each word. */

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

  /* Free up the memory allocated with new_inputstruct, and
     close the open file.  This is not necessary in this program,
     since we are exiting anyway, but I just want to show how you free it up. */

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

- Bu prosedürler, girdi dosyalarını işlemek için çok uygundur. Örneğin, aşağıdaki program (src/printwords.c'de) bir girdi dosyasındaki her kelimeyi satır numarasının başına gelecek şekilde yazdırır.

```
UNIX> bin/printwords txt/rex-1.txt
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>
```

stdder

- C programlama dilinde, standart çıktı olarak da bilinen farklı dosya tanımlayıcıları vardır.
- Standart giriş için stdin,
- standart çıkış için stdout ve
- hata mesajı çıkışı için stderr olmak üzere 3 standart I/O yöntemi vardır.

```
#include <stdio.h> int main() {
fprintf(stdout,"This is message 1\n");
fprintf(stderr,"This is message 2\n");
fprintf(stdout,"This is message 3\n");
return(0); }
```

```
This is message 1
This is message 2
This is message 3
```

stdin

```
#include <stdio.h>
int main( ) {

  char a[100];

  fprintf( "Enter a string :");
  fscanf(stdin,"%s", a);

  fprintf( stdout,"\nYou entered the following string: %s ", a);

  fprintf(stdout,“\n”);
  return 0;

}
```

fgets

```
char *fgets(char *str, int n, FILE *stream)
```

- Parametreler
- **str** - Bu, okunan string'in depolandığı bir karakter dizisinin işaretçisidir.
- **n** – Bu, okunacak maksimum karakter sayısıdır (son boş karakter dahil). Genellikle str olarak geçirilen dizinin uzunluğu kullanılır.
- **stream** – Bu, karakterlerin okunduğu akışı tanımlayan bir FILE nesnesinin işaretçisidir.

Fields

- Fields kitaplığıyla ilgili dikkat edilmesi gereken önemli bir nokta, malloc()'un çağrıldığı tek zamanın new_inputstruct() sırasında olmasıdır.
- Get_line() sadece IS yapısının alanlarını doldurur --- bellek tahsisi gerçekleştirmez.
- Bu, bir satırı veya alanı depolamak ve sonraki get_line() çağrısının üzerine yazılmasını istemiyorsanız, genellikle strdup() ile bunun bir kopyasını oluşturmanız gerektiği anlamına gelir.
- Bu çok önemli, genel olarak fgets() ile yapılan en yaygın hata, bir kopyaya ihtiyaç duyduklarında bir kopya çıkarmamaktır.
- İşaretçiler ve malloc() konusunda size yardımcı olacak bu hatayı burada göstereceğiz.

Fields

- Amacımız, standart girdinin son n satırını basan program kuyruğunu yazmak olacaktır.
- n'nin değeri varsayılan olarak 10'dur, ancak bunu komut satırında belirtebilmelisiniz.
- Fields kitaplığını kullanarak son 10 satırı yazdırmaya çalışacak olan src/tail10-bad.c yazarak başlayalım.
- Bu, yukarıda bahsedilen yaygın hatayı gösterecektir.
- İşte oldukça basit olan kod. 10 karakterden oluşan bir * dizimiz olacak, her satırı okuduğumuz zaman basitçe is->text1 olarak ayarlayacağız:

src/tail10-bad.c,

```c
/* A buggy program to print the last 10 lines of standard input. */

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

int main(int argc, char **argv)
{
  IS is;
  int i, n;
  char *lines[10];    /* This array will hold the last 10 lines of standard input. */

  /* Read the lines of standard input, and only keep the last ten. */

  is = new_inputstruct(NULL);
  n = 0;
  while (get_line(is) >= 0) {
    lines[n%10] = is->text1;        /* This is the bad line -- it doesn't copy the string. */
    n++;
  }

  /* Print the last 10 lines, or fewer if there are fewer lines.
     Remember that is->text1 has a newline at the end. */

  i = (n >= 10) ? (n-10) : 0;                     /* This is the line number of the 10th line from the end. */
  for ( ; i < n; i++) printf("%s", lines[i%10]);   /* Print this line to the last line. */

  return 0;
}
```

src/tail10-bad.c,

```
UNIX> cat txt/tail-input-15.txt
     1  Elijah Christian Shatterproof
     2  Cameron Ostracod
     3  Ryan Sargent
     4  Christopher Tempest
     5  Aiden Circumferential
     6  Carson Carcass
     7  Caroline Jazz
     8  Molly Jade
     9  Jordan Equivalent MD
    10  Aaron Nagging
    11  Isaac Bandwidth
    12  Leah Bulk
    13  Victoria Glutamate
    14  Lucas Workmen
    15  Sofia Godlike
UNIX>
```

```
UNIX> bin/tail10-bad < txt/tail-input-15.txt
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
UNIX>
```

- Cat komutu, Herhangi bir seçenek belirtmeden komutun kendisi bir dosyanın içeriklerini okuyacak ve konsolda gösterecektir.
- Cat komutu, standart çıktı ekranında dosyalar oluşturmanıza, birleştirmenize veya bastırmanıza izin verir

src/tail10-bad-print.c,

```
UNIX> bin/tail10-bad-print < txt/tail-input-15.txt
I have set lines[0] to 0x7fc014002614, which is currently      1        Elijah Christian Shatterproof
I have set lines[1] to 0x7fc014002614, which is currently      2        Cameron Ostracod
I have set lines[2] to 0x7fc014002614, which is currently      3        Ryan Sargent
I have set lines[3] to 0x7fc014002614, which is currently      4        Christopher Tempest
I have set lines[4] to 0x7fc014002614, which is currently      5        Aiden Circumferential
I have set lines[5] to 0x7fc014002614, which is currently      6        Carson Carcass
I have set lines[6] to 0x7fc014002614, which is currently      7        Caroline Jazz
I have set lines[7] to 0x7fc014002614, which is currently      8        Molly Jade
I have set lines[8] to 0x7fc014002614, which is currently      9        Jordan Equivalent MD
I have set lines[9] to 0x7fc014002614, which is currently     10        Aaron Nagging
I have set lines[0] to 0x7fc014002614, which is currently     11        Isaac Bandwidth
I have set lines[1] to 0x7fc014002614, which is currently     12        Leah Bulk
I have set lines[2] to 0x7fc014002614, which is currently     13        Victoria Glutamate
I have set lines[3] to 0x7fc014002614, which is currently     14        Lucas Workmen
I have set lines[4] to 0x7fc014002614, which is currently     15        Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
    15  Sofia Godlike
UNIX>
```

```
printf("I have set lines[%d] to 0x%lx, which is currently %s",
        n%10, (unsigned long) (lines[n%10]), lines[n%10]);
```

src/tail10-memory-leak.c,

```
  is = new_inputstruct(NULL);
  n = 0;
  while (get_line(is) >= 0) {
    lines[n%10] = strdup(is->text1);       /* This is the only change - we call strdup(). */
    n++;
  }
}
```

```
UNIX> bin/tail10-memory-leak < txt/tail-input-15.txt
     6  Carson Carcass
     7  Caroline Jazz
     8  Molly Jade
     9  Jordan Equivalent MD
    10  Aaron Nagging
    11  Isaac Bandwidth
    12  Leah Bulk
    13  Victoria Glutamate
    14  Lucas Workmen
    15  Sofia Godlike
UNIX>
```

- Basit düzeltme strdup() kullanmaktır.
- Bu, satırın bir kopyası için bellek ayıracak ve ardından satırı kopyalayacaktır.
- Kod src/tail10-memory-leak.c'de, adından da anlayabileceğiniz gibi, kendine ait bazı sorunları olacak. Tek değişiklik, artık satırları[n%10] is->text1'e atamamız, bunun yerine strdup() ile bir kopya oluşturmamızdır.

src/tail10-memory-leak.c,

- Aslında, çoğu girişte gayet iyi çalışır.
- Ancak adından da anlaşılacağı gibi bir bellek sızıntısı var.
- n, 10'dan büyük veya eşit olduğunda, strdup() satırı şu anda [n%10] satırlarında bulunan işaretçinin üzerine yazar ve işaretçi sonsuza dek kaybolur.
- Bununla birlikte, işaret ettiği bellek yine de tahsis edilmiştir ve programdan çıkılana kadar yeniden tahsis edilmeyecektir.
- Bellek sızıntısının tam tanımı budur.
- Bunu çok sayıda satır içeren bir girdi üzerinde çalıştırırsak, programın bellek kullanımı patlayacak ve sonunda makineniz durma noktasına gelecek ve/veya strdup() başarısız olduğunda sonlanacaktır.

src/tail10-memory-leak.c

- UNIX> echo "" | awk '{ while (1) print "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" }' | bin/tail10-memory-leak &
- awk script'i X'lerle sonsuz sayıda satır yazdırır

```
UNIX> echo "" | awk '{ while (1) print "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" }' | bin/tail10-memory-leak  &
```

Then, I take a look at how the program is running with **top**:

```
UNIX> top
PID    COMMAND      %CPU  TIME     #TH   #WQ  #PORT MEM    PURG   CMPRS  PGRP  PPID  STATE
.....  All my running processes ....
```

Here is how the program is running at 6 seconds, 30 seconds and 60 seconds:

```
84909  tail10-memor 99.8  00:06.13 1/1   0    12    745M+  0B     0B     84907 79428 running
...
84909  tail10-memor 99.8  00:30.87 1/1   0    12    3807M+ 0B     0B     8490  79428 running
...
84909  tail10-memor 99.8  01:00.83 1/1   0    12    7469M+ 0B     0B     84907 79428 running
```

src/tail10-good.c

- Öyleyse düzeltelim. strdup() malloc()'u çağırırız, böylece artık dizeye ihtiyacımız kalmadığında ve işaretçinin üzerine yazmak üzereyken dizeyi serbest bırakırız.

```
is = new_inputstruct(NULL);
n = 0;
while (get_line(is) >= 0) {
  if (n >= 10) free(lines[n%10]);     /* This line prevents the memory leak. */
  lines[n%10] = strdup(is->text1);
  n++;
}
```

```
PID    COMMAND      %CPU  TIME     #TH   #WQ  #PORT MEM    PURG   CMPRS  PGRP  PPID  STATE
85101  tail10-good  99.7  00:07.44 1/1   0    12    492K   0B     0B     85099 79428 running
...
85101  tail10-good  99.9  00:30.15 1/1   0    12    492K   0B     0B     85099 79428 running
...
85101  tail10-good  99.9  01:00.11 1/1   0    12    492K   0B     0B     85099 79428 running
```

tail10-good print

```
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Fields/bin$ ./tail10-good < ../txt/tail-input-15.txt
I have set lines[0] to 0x5556ea6719b0, which is currently      1        Elijah Christian Shatterproof
I have set lines[1] to 0x5556ea671df0, which is currently      2        Cameron Ostracod
I have set lines[2] to 0x5556ea671e20, which is currently      3        Ryan Sargent
I have set lines[3] to 0x5556ea671e40, which is currently      4        Christopher Tempest
I have set lines[4] to 0x5556ea671e70, which is currently      5        Aiden Circumferential
I have set lines[5] to 0x5556ea671ea0, which is currently      6        Carson Carcass
I have set lines[6] to 0x5556ea671ec0, which is currently      7        Caroline Jazz
I have set lines[7] to 0x5556ea671ee0, which is currently      8        Molly Jade
I have set lines[8] to 0x5556ea671f00, which is currently      9        Jordan Equivalent MD
I have set lines[9] to 0x5556ea671f30, which is currently     10        Aaron Nagging
I have set lines[0] to 0x5556ea671f50, which is currently     11        Isaac Bandwidth
I have set lines[1] to 0x5556ea671f70, which is currently     12        Leah Bulk
I have set lines[2] to 0x5556ea671df0, which is currently     13        Victoria Glutamate
I have set lines[3] to 0x5556ea671e20, which is currently     14        Lucas Workmen
I have set lines[4] to 0x5556ea671f90, which is currently     15        Sofia Godlike
     6  Carson Carcass
     7  Caroline Jazz
     8  Molly Jade
     9  Jordan Equivalent MD
    10  Aaron Nagging
    11  Isaac Bandwidth
    12  Leah Bulk
    13  Victoria Glutamate
    14  Lucas Workmen
    15  Sofia Godlike
```

tailanyf

Komut satırında satır sayısını belirttiğiniz tail'in genel versiyonudur.

Bu program, alışmanız gereken birkaç şeyi göstermektedir:

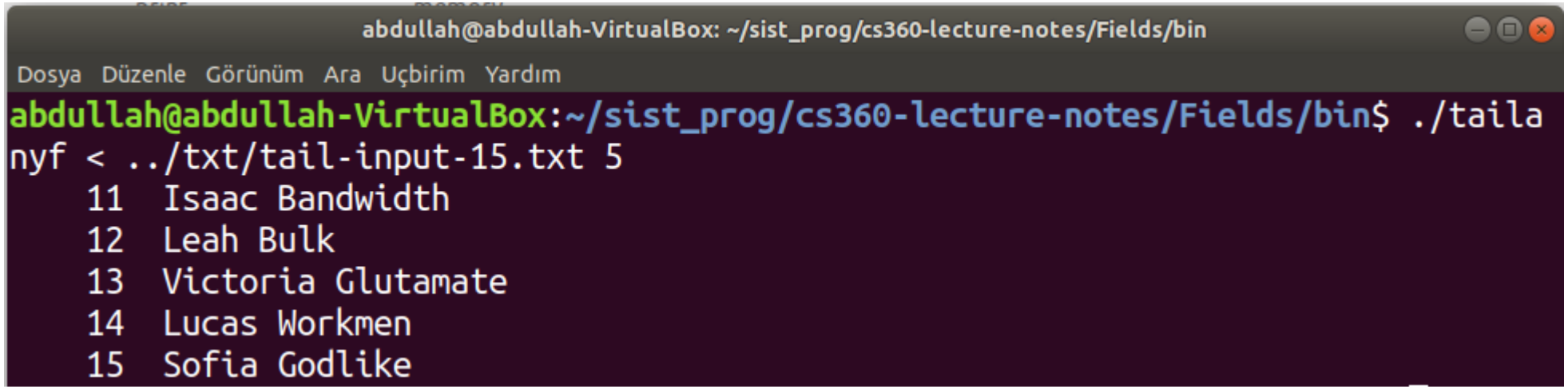
1. Bir string’i tamsayıya dönüştürmek ve çalışıp çalışmadığını test etmek için sscanf() işlevini kullanma.
2. Çalışma zamanına kadar boyutu bilinmeyen bir diziyi ayırmak için malloc() kullanma.
3. Fields kitaplığını kullanma ve bir dizenin bir kopyasını depolamak için strdup() kullanma.

tailanyf

```c
/* This program is more like tail -- it takes the number of lines, n,
   as a command line argument, and prints the last n lines of standard input. */

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

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

  /* Error check the command line. */

  if (argc != 2) { fprintf(stderr, "usage: tailany1 n\n"); exit(1); }
  if (sscanf(argv[1], "%d", &n) == 0 || n <= 0) {
    fprintf(stderr, "usage: tailany1 n\n");
    fprintf(stderr, "       bad n: %s\n", argv[1]);
    exit(1);
  }

  /* Allocate the array */

  lastn = (char **) malloc(sizeof(char *)*n);
  if (lastn == NULL) { perror("malloc"); exit(1); }
```

```c
  /* Allocate the array */

  lastn = (char **) malloc(sizeof(char *)*n);
  if (lastn == NULL) { perror("malloc"); exit(1); }

  /* Allocate the IS */

  is = new_inputstruct(NULL);
  if (is == NULL) { perror("stdin"); exit(1); }

  /* Read the input */

  nlines = 0;
  while (get_line(is) >= 0) {
    if (nlines >= n) free(lastn[nlines%n]);     /* Prevent the memory leak. */
    lastn[nlines%n] = strdup(is->text1);
    nlines++;
  }

  /* Print the last n lines */

  i = (nlines < n) ? 0 : nlines-n;
  for ( ; i < nlines; i++) {
    printf("%s", lastn[i%n]);
  }

  /* Don't bother freeing stuff when you're just exiting anyway. */

  return 0;
}
```

tailanyf print



pipe_inputstruct()

- Bu, popen() ile açılan bir kanaldan(pipe) okumanızı sağlar.
- src/pipetest.c programı, src dizinindeki tüm .c dosyalarındaki satır sayısını saymak için pipe_inputstruct() yöntemini kullanır.
- Bunu, "cat src/*.c" standart çıktısını bir inputstruct'a almak için pipe_inputstruct() kullanarak yapar:

pipe_inputstruct()

```c
/* pipetest.c counts the number of lines in all the .c files in the
   src directory.  It does this by using pipe_inputstruct to get
   the standard output of the cat command into an inputstruct */

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

int main()
{
  IS is;
  int nlines;

  is = pipe_inputstruct("cat src/*.c");
  if (is == NULL) { perror("cat src/*.c"); exit(1); }

  nlines = 0;
  while (get_line(is) >= 0) nlines++;

  printf("# lines in src/*.c: %d\n", nlines);

  return 0;
}
```

pipetest print