Διάλεξη 18 - Ταξινόμηση και Δεδομένα Εισόδου #2 (Αρχεία)

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στον Προγραμματισμό

Θανάσης Αυγερινός / Τάκης Σταματόπουλος

Ανακοινώσεις / Διευκρινίσεις

1. Πόσος χώρος δεσμεύεται από την δήλωση int array[5][10];

    a. Ποιο είναι το sizeof(array[3]);

        10 * sizeof(int)

2. Δεν αναγνωρίζω την πολυπλοκότητα log(n) - τι κάνω;

A. Ελέγχω αν μπορώ να "σπάσω" το γενικό μου πρόβλημα σε μικρότερα υποπροβλήματα παρεμφερούς μεγέθους και μετά αν χρειάζεται να λύσω μόνο ένα από αυτά.
B. Ελέγχω πως συμπεριφέρεται το πρόγραμμά μου καθώς αυξάνεται το μέγεθος N της εισόδου. Αν οι επαναλήψεις αυξάνονται γραμμικά καθώς το N αυξάνεται εκθετικά, έχω logn πολυπλοκότητα. Π.χ., N = 10 -> 1 επανάληψη, N = 100 -> 2 επαναλήψεις, N = 1000000 -> 6 επαναλήψεις κοκ.

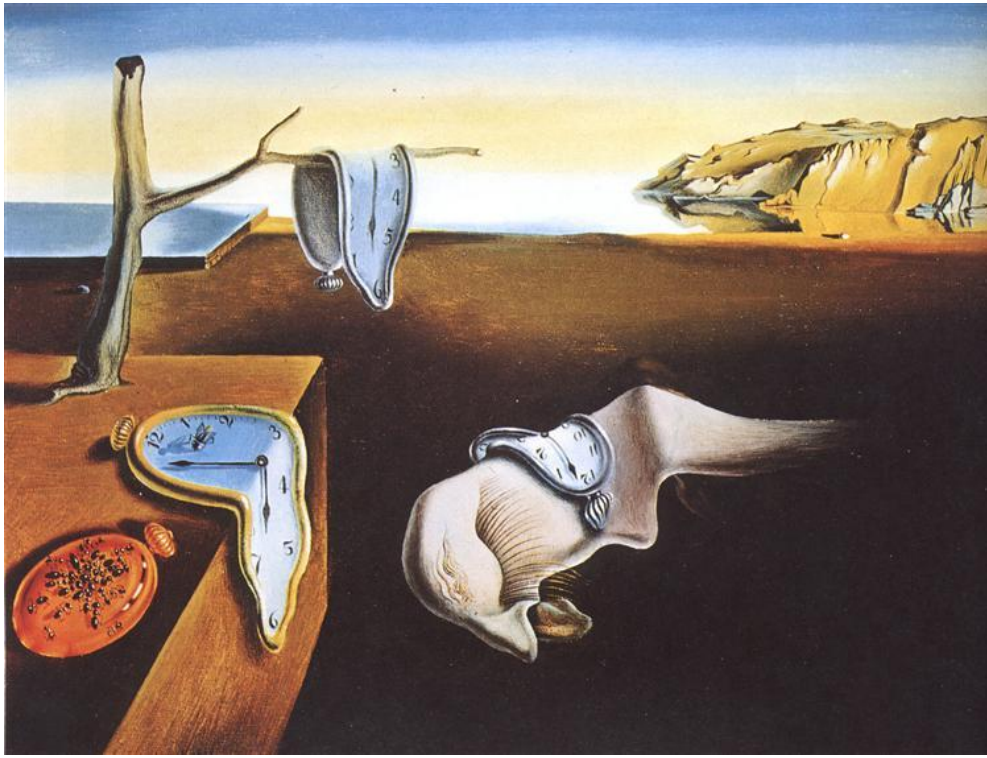
Την Προηγούμενη Φορά

- Αλγόριθμοι Αναζήτησης

  (Search Algorithms)

  - Γραμμική Αναζήτηση (Linear Search)
  - Δυαδική Αναζήτηση (Binary Search)
- Αλγόριθμοι Ταξινόμησης

  (Sorting Algorithms)



Σήμερα

- Αλγόριθμοι Ταξινόμησης

  (Sorting Algorithms)
- Δεδομένα Εισόδου #2
  - Λίγο παραπάνω για την scanf
  - Αρχεία (Files) και ρεύματα

    δεδομένων (data streams)
  - Συναρτήσεις χειρισμού αρχείων

Το Instagram έχει 2 δισεκατομμύρια χρήστες σε έναν πίνακα ακεραίων (έστω ένας ακέραιος ανά χρήστη). Πόσα βήματα (χρονική πολυπλοκότητα) χρειάζεστε για να βρείτε αν ο χρήστης 424242 βρίσκεται στον πίνακα;

Αλγόριθμοι Ταξινόμησης (Sorting Algorithms)

1. Bubblesort
2. Selection Sort
3. Insertion Sort
4. Merge Sort
5. Quicksort

Γράψτε μια συνάρτηση swap που ανταλλάζει δύο ακεραίους

```c
#include <stdio.h>
int main() {
  int a = 100, b = 200;
  printf("%d %d\n", a, b);
  swap( ... );
  printf("%d %d\n", a, b);
  return 0;
}
```

Γράψτε μια συνάρτηση swap που ανταλλάζει δύο ακεραίους

```c
#include <stdio.h>
void swap(int *a, int *b) {
  int tmp = *a;
  *a = *b;
  *b = tmp;
}
int main() {
  int a = 100, b = 200;
  printf("%d %d\n", a, b);
  swap(&a, &b);
  printf("%d %d\n", a, b);
  return 0;
}
```

Ταξινόμηση Επιλογής (Selection Sort)

```
void selection_sort(int n, int *x) {
  int i, j, min;
  for (i = 1 ; i <= n - 1 ; i++) {
    min = i - 1;
    for (j = i ; j <= n - 1 ; j++)
      if (x[j] < x[min])
        min = j;
    swap(&x[i-1], &x[min]);
  }
}
```

Ταξινόμηση Επιλογής (Selection Sort)

```
void selection_sort(int n, int *x) {
  int i, j, min;
  for (i = 1 ; i <= n - 1 ; i++) {
    min = i - 1;
    for (j = i ; j <= n - 1 ; j++)
      if (x[j] < x[min])
        min = j;
    swap(&x[i-1], &x[min]);
  }
}
```

Χρόνος: $O(n^2)$
Χώρος: $O(1)$

Ταξινόμηση Εισαγωγής (Insertion Sort)

```
void insertion_sort(int n, int *x) {
  int i, j;
  for (i = 1 ; i <= n - 1 ; i++) {
    j = i - 1;
    while (j >= 0 && x[j] > x[j+1]) {
      swap(&x[j], &x[j+1]);
      j--;
    }
  }
}
```

Ταξινόμηση Εισαγωγής (Insertion Sort)

```
void insertion_sort(int n, int *x) {
  int i, j;
  for (i = 1 ; i <= n - 1 ; i++) {
    j = i - 1;
    while (j >= 0 && x[j] > x[j+1]) {
      swap(&x[j], &x[j+1]);
      j--;
    }
  }
}
```

Χρόνος: $O(n^2)$
Χώρος: $O(1)$

Ταξινόμηση Φυσαλίδας (Bubblesort)

```
void bubblesort(int n, int *x) {
  int i, j;
  for (i = 1 ; i <= n - 1 ; i++)
    for (j = n - 1 ; j >= i ; j--)
      if (x[j-1] > x[j])
        swap(&x[j-1], &x[j]);
}
```

Ταξινόμηση Φυσαλίδας (Bubblesort)

```
void bubblesort(int n, int *x) {
  int i, j;
  for (i = 1 ; i <= n - 1 ; i++)
    for (j = n - 1 ; j >= i ; j--)
      if (x[j-1] > x[j])
        swap(&x[j-1], &x[j]);
}
```

Χρόνος: $O(n^2)$
Χώρος: $O(1)$

Ταξινόμηση Συγχώνευσης (Merge Sort)

Η ταξινόμηση συγχώνευσης (merge sort) είναι ένας αλγόριθμος divide and conquer (διαίρει και βασίλευε) που έχει θεωρητικά την καλύτερη πολυπλοκότητα. Ο αλγόριθμος έχει δύο βήματα:

1. Χώρισε τον πίνακα σε δύο υποπίνακες
    a. κάλεσε ταξινόμηση συγχώνευσης στους υποπίνακες
2. Συγχώνευσε τα στοιχεία των δύο ταξινομημένων υποπινάκων

Ταξινόμηση Συγχώνευσης (Merge Sort)

```
void merge_sort(int *array, int left, int right) {
    if (left < right) {
        int middle = left + (right - left) / 2;
        merge_sort(array, left, middle);
        merge_sort(array, middle + 1, right);
        merge(array, left, middle, right);
    }
}
```

Ταξινόμηση Συγχώνευσης (Merge Sort)

```
void merge(int *x, int l, int m, int r) {
    int i, j, k, n1 = m - l + 1, n2 = r - m;
    int left[n1], right[n2];
    for (i = 0; i < n1; i++) left[i] = x[l + i];
    for (j = 0; j < n2; j++) right[j] = x[m + 1 + j];
    i = 0; j = 0; k = l;
    while (i < n1 && j < n2) {
        if (left[i] <= right[j]) x[k++] = left[i++];
        else x[k++] = right[j++];
    }
    while (i < n1) x[k++] = left[i++];
    while (j < n2) x[k++] = right[j++];
}
```

Ταξινόμηση Συγχώνευσης (Merge Sort)

```
void merge(int *x, int l, int m, int r) {
    int i, j, k, n1 = m - l + 1, n2 = r - m;
    int left[n1], right[n2];
    for (i = 0; i < n1; i++) left[i] = x[l + i];
    for (j = 0; j < n2; j++) right[j] = x[m + 1 + j];
    i = 0; j = 0; k = l;
    while (i < n1 && j < n2) {
        if (left[i] <= right[j]) x[k++] = left[i++];
        else x[k++] = right[j++];
    }
    while (i < n1) x[k++] = left[i++];
    while (j < n2) x[k++] = right[j++];
}
```

Χρόνος: O(nlogn)
Χώρος: O(n)

Thanks John von Neumann

Ταχυταξινόμηση (Quicksort)

Η ταξινόμηση ταχυταξινόμηση (quicksort) είναι ένας αλγόριθμος divide and conquer (διαίρει και βασίλευε) που είναι **ιδιαίτερα δημοφιλής**. Ο αλγόριθμος έχει τρία βήματα:

1. Διάλεξε (έστω τυχαία) το στοιχείο διαμέρισης του πίνακα (pivot element)
2. Διαμέρισε τον πίνακα σε δύο υποπίνακες - αριστερά έχει τα στοιχεία που είναι μικρότερα του pivot και δεξιά τα στοιχεία που είναι μεγαλύτερα
3. Τρέξε ταχυταξινόμηση για τους δύο υποπίνακες

Ταχυταξινόμηση (Quicksort)

```
void quicksort (int *x, int lower, int upper) {
   if (lower < upper) {
      int pivot = x[(lower + upper) / 2];
      int i, j;
      for (i = lower, j = upper; i <= j;) {
        while (x[i] < pivot) i++;
        while (x[j] > pivot) j--;
        if (i <= j) swap(&x[i++], &x[j--]);
         }
      quicksort(x, lower, j);
      quicksort(x, i, upper);
   }
 }
```

Ταχυταξινόμηση (Quicksort)

```
void quicksort (int *x, int lower, int upper) {
   if (lower < upper) {
      int pivot = x[(lower + upper) / 2];
      int i, j;
      for (i = lower, j = upper; i <= j;) {
        while (x[i] < pivot) i++;
        while (x[j] > pivot) j--;
        if (i <= j) swap(&x[i++], &x[j--]);
         }
      quicksort(x, lower, j);
      quicksort(x, i, upper);
   }
}
```

Χρόνος: $O(n^2)$ (worst case), O(nlogn) (average case)
Χώρος: O(n) (εδώ) - γίνεται και σε O(logn)

Υλοποιημένη στην συνάρτηση `qsort` της stdlib.h

Thanks Tony Hoare

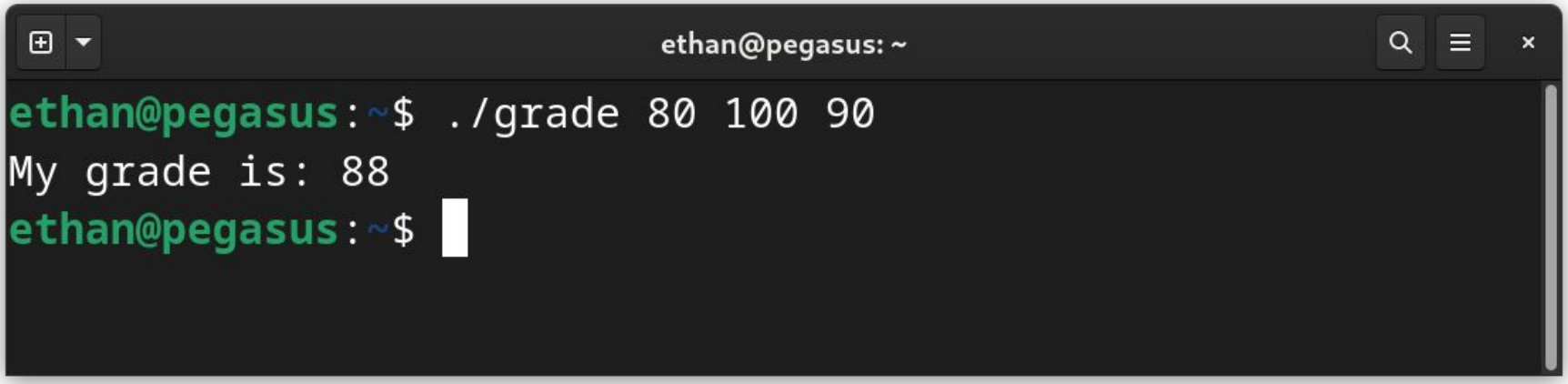
Δεδομένα Εισόδου #2 (Αρχεία)

Δεδομένα Εισόδου και Εξόδου (Input and Output Data)

Δεδομένα Εισόδου (Input Data)

Τα δεδομένα εισόδου (input data) είναι μια σειρά από χαρακτήρες (bytes) τα οποία ο χρήστης δίνει στο πρόγραμμα. Υπάρχουν 4 μέθοδοι να εισάγουμε δεδομένα:

1. Ορίσματα στην γραμμή εντολών



Δεδομένα Εισόδου (Input Data) - 2/4

Τα δεδομένα εισόδου (input data) είναι μια σειρά από χαρακτήρες (bytes) τα οποία ο χρήστης δίνει στο πρόγραμμα. Υπάρχουν 4 μέθοδοι να εισάγουμε δεδομένα:

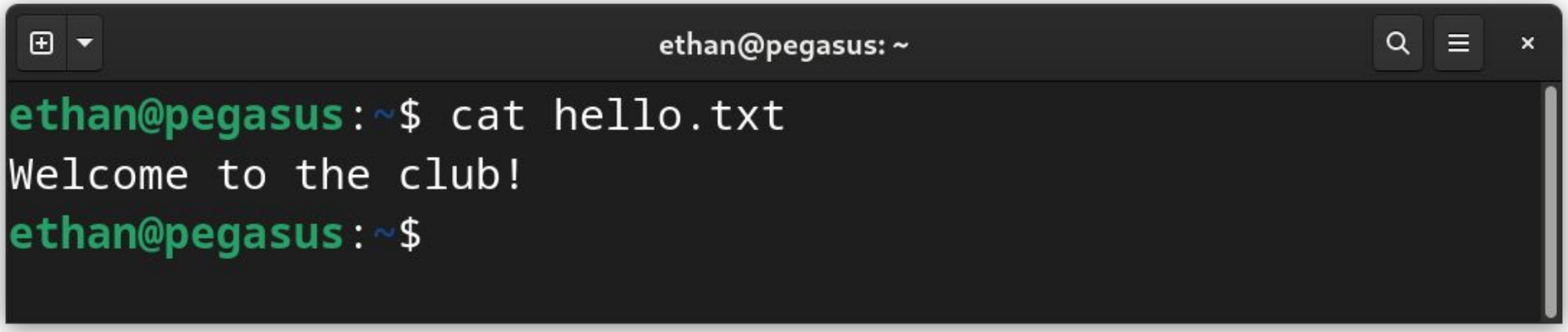
2. Γράφοντας κείμενο στην πρότυπη είσοδο (standard input ή stdin) συνήθως με το πληκτρολόγιο

```
thanassis@Plato:~$ ./aliquot
Please give the number to start the aliquot sequence from: 138
Provide the max aliquot length to look for (0 for unlimited): 0
Do you want to print the full sequence ('f') or just the length ('l')? l
Length of aliquot sequence: 178
thanassis@Plato:~$
```

Δεδομένα Εισόδου (Input Data) - 3/4

Τα δεδομένα εισόδου (input data) είναι μια σειρά από χαρακτήρες (bytes) τα οποία ο χρήστης δίνει στο πρόγραμμα. Υπάρχουν 4 μέθοδοι να εισάγουμε δεδομένα:

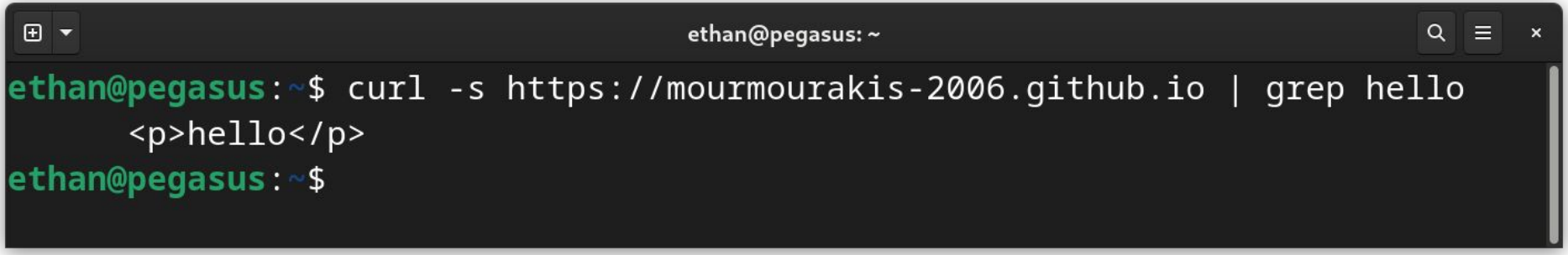
3. Διαβάζοντας αρχεία από το σύστημα αρχείων (σήμερα!)



Δεδομένα Εισόδου (Input Data) - 4/4

Τα δεδομένα εισόδου (input data) είναι μια σειρά από χαρακτήρες (bytes) τα οποία ο χρήστης δίνει στο πρόγραμμα. Υπάρχουν 4 μέθοδοι να εισάγουμε δεδομένα:

4. Διαβάζοντας από το δίκτυο ή άλλες πηγές - π.χ., User Interface (σε επόμενα εξάμηνα)



Δεδομένα Εισόδου (Input Data)

Τα δεδομένα εισόδου (input data) είναι μια σειρά από χαρακτήρες (bytes) τα οποία ο χρήστης δίνει στο πρόγραμμα. Υπάρχουν 4 μέθοδοι να εισάγουμε δεδομένα:

1. Ορίσματα στην γραμμή εντολών ✅
2. Γράφοντας κείμενο στην πρότυπη είσοδο (standard input ή stdin) ✅
3. Διαβάζοντας αρχεία από το σύστημα αρχείων (σήμερα!) ✅
4. Διαβάζοντας από το δίκτυο ή άλλες πηγές (άλλα εξάμηνα)

Χρήση της συνάρτησης `scanf` - Άλλοι τύποι ορισμάτων

Τι κάνει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>
#include <math.h>
int main() {
  double d1, d2;
  printf("Gimme two doubles: ");
  scanf("%lf %lf", &d1, &d2);
  printf("Hypotenuse: %.1f\n", sqrt(d1 * d1 + d2 * d2));
  return 0;
}
```

Χρήση της συνάρτησης `scanf` - Άλλοι τύποι ορισμάτων

Τι κάνει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>
#include <math.h>
int main() {
  double d1, d2;
  printf("Gimme two doubles: ");
  scanf("%lf %lf", &d1, &d2);
  printf("Hypotenuse: %.1f\n", sqrt(d1 * d1 + d2 * d2));
  return 0;
}
```

```
$ ./hypotenuse
Gimme two numbers: 3.0 4.0
Result: 5.0
```

Χρήση της συνάρτησης `scanf` - Άλλοι τύποι ορισμάτων

Τι κάνει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>
int main() {
  char message[7];
  printf("Say something: ");
  scanf("%s", message);
  printf("%s\n", message);
  return 0;
}
```

Χρήση της συνάρτησης `scanf` - Άλλοι τύποι ορισμάτων

Τι κάνει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>
int main() {
  char message[7];
  printf("Say something: ");
  scanf("%s", message);
  printf("%s\n", message);
  return 0;
}
```

```
$ ./message
Say something: hello!
hello!
```

Δεν βάλαμε & πριν την μεταβλητή message. Πως και λειτουργεί;

Χρήση της συνάρτησης `scanf` - Άλλοι τύποι ορισμάτων

Τι κάνει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>
int main() {
  char message[7];
  printf("Say something: ");
  scanf("%s", message);
  printf("%s\n", message);
  return 0;
}
```

```
$ ./message
Say something: hello!
hello!
```

Μπορεί να πάει κάτι στραβά εδώ;

Χρήση της συνάρτησης `scanf` - Άλλοι τύποι ορισμάτων

Τι κάνει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>
int main() {
  char message[7];
  printf("Say something: ");
  scanf("%s", message);
  printf("%s\n", message);
  return 0;
}
```

```
$ ./message
Say something: Houston,
we've had a problem here.
Houston,
Segmentation fault
```

Προσοχή! Δεν υπάρχει κανένας έλεγχος ότι δεν θα διαβαστούν περισσότεροι χαρακτήρες από 6

Χρήση της συνάρτησης `scanf` - Άλλοι τύποι ορισμάτων

Τι κάνει το παρακάτω πρόγραμμα;

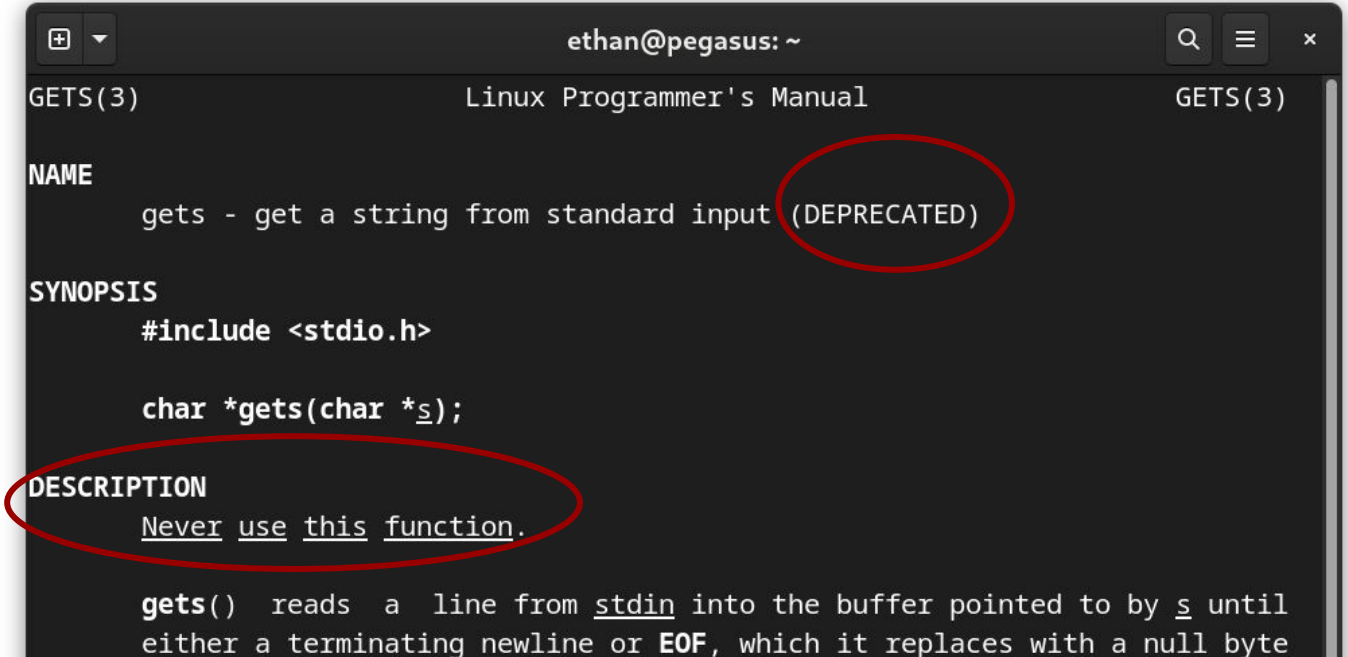
```c
#include <stdio.h>
int main() {
  char message[7];
  printf("Say something: ");
  scanf("%6s", message);
  printf("%s\n", message);
  return 0;
}
```

```
$ ./message
Say something: Houston,
we've had a problem here.
Housto
```

Μπορούμε να θέσουμε περιορισμό στους πόσους χαρακτήρες θα διαβαστούν

Η συνάρτηση `gets`

Η συνάρτηση `gets` συμπεριφέρεται παρόμοια με την `scanf("%s")` και γενικά αποφεύγεται για λόγους ασφαλείας.



Θέλω οπωσδήποτε να χρησιμοποιήσω scanf; %ms since C11

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char ** argv) {
  char *string; int items_read;
  items_read = scanf("%ms", &string);
  if (items_read != 1) {
      fprintf(stderr, "No matching characters\n");
      return 1;
  }
  printf("read the following string: %s\n", string);
  // don't forget to free!
  free(string);
  return 0;
}
```

Χειρισμός Αρχείων (File Handling)

Filesystem (Σύστημα Αρχείων)

Αρχείο (File) είναι ένας πόρος για να καταγράφουμε δεδομένα σε έναν υπολογιστή. Συνήθως αποθηκεύεται στην δευτερεύουσα μνήμη (π.χ., σκληρός δίσκος).

Στο Linux σχεδόν τα πάντα είναι ένα αρχείο.

Το περιεχόμενο του αρχείου είναι ένας πίνακας από χαρακτήρες `char bytes[]`

Κάθε αρχείο:

- Έχει ένα όνομα (**filename/basename**)
- Βρίσκεται μέσα σε ένα συγκεκριμένο φάκελο (**directory/folder**)
- Έχει ένα πλήρες μονοπάτι (**filepath**) που καθορίζει που βρίσκεται το αρχείο.

Παράδειγμα: το filepath ενός αρχείου είναι `/home/users/thanassis/documents/students.txt`, ο φάκελος μέσα στον οποίο βρίσκεται αυτό το αρχείο είναι ο `/home/users/thanassis/documents` ενώ το όνομα του αρχείου είναι `students.txt`. Το `.txt` στο τέλος του ονόματος λέγεται επέκταση (**extension**) και συνήθως να περιγράφει τον τύπο του αρχείου.

Ο τύπος FILE

Ο τύπος FILE ορίζεται στην κεφαλίδα stdio.h και μας επιτρέπει να αναφερόμαστε σε αρχεία που άνοιξε το πρόγραμμά μας.

```
#include <stdio.h>

int main() {

  printf("Size of FILE type: %zu\n", sizeof(FILE));

  return 0;

}
```

```
$ ./filedef
216
```

216 bytes σε ένα σύστημα Debian! Τι περιέχει εντός; Πολλά και διάφορα ίσως το συζητήσουμε άλλη φορά

Η συνάρτηση `fopen`

Η συνάρτηση `fopen` επιτρέπει στο πρόγραμμά μας να ανοίξει ένα αρχείο και να επιστρέψει έναν δείκτη σε FILE. Επιστρέφει NULL αν αποτύχει να ανοίξει το αρχείο για οποιοδήποτε λόγο. Η δήλωση της συνάρτησης είναι

```
FILE *fopen(const char *restrict pathname, const char *restrict mode);
```

pathname: το μονοπάτι που αντιστοιχεί στο αρχείο

mode: με ποιο τρόπο να ανοίξουμε το αρχείο (διάβασμα ή γράψιμο;)

DESCRIPTION

The **fopen**() function opens the file whose name is the string pointed to by pathname and associates a stream with it.

The argument mode points to a string beginning with one of the following sequences (possibly followed by additional characters, as described below):

r Open text file for reading. The stream is positioned at the beginning of the file.

r+ Open for reading and writing. The stream is positioned at the beginning of the file.

w Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.

w+ Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.

a Open for appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.

a+ Open for reading and appending (writing at end of file). The file is created if it does not exist. Output is always appended to the end of the file. POSIX is silent on what the initial read position is when using this mode. For glibc, the initial file position for reading is at the beginning of the file, but for Android/BSD/MacOS, the initial file position for reading is at the end of the file.

Παράδειγμα: Άνοιγμα Αρχείων

```c
#include <stdio.h>

int main() {

    FILE *fileToRead, *fileToWrite;

    fileToRead = fopen("input.txt", "r");

    if (!fileToRead) {

        return 1;

    }

    fileToWrite = fopen("output.txt", "w");

    if (!fileToWrite) {

        return 1;

    }

    return 0;

}
```

Ελέγχουμε πάντα το αποτέλεσμα της fopen αν είναι NULL. Για ποιο λόγο μπορεί να αποτύχει;

Παράδειγμα: Άνοιγμα Αρχείων

```c
#include <stdio.h>

int main() {

    FILE *fileToRead, *fileToWrite;

    fileToRead = fopen("input.txt", "r");

    if (!fileToRead) {

        return 1;

    }

    fileToWrite = fopen("output.txt", "w");

    if (!fileToWrite) {

        return 1;

    }

    return 0;

}
```

Ελέγχουμε πάντα το αποτέλεσμα της fopen αν είναι NULL. Για ποιο λόγο μπορεί να αποτύχει;

1. Το αρχείο δεν υπάρχει
2. Ο φάκελος δεν υπάρχει
3. Δεν έχουμε ελεύθερο δίσκο για να γράψουμε αρχείο!
4. Δεν έχουμε δικαιώματα να φτιάξουμε αρχείο
5. Έχουμε ανοίξει τον μέγιστο επιτρεπτό αριθμό αρχείων
6. …

Η συνάρτηση `fclose`

Η συνάρτηση `fclose` μας επιτρέπει να κλείσουμε ένα αρχείο. Επιστρέφει 0 αν επιτύχει ή EOF αν αποτύχει. Η δήλωση της συνάρτησης είναι

```
int fclose(FILE *stream);
```

Πάντα κλείνουμε τα αρχεία που άνοιξε το πρόγραμμά μας αφού τελειώσουμε με την χρήση τους.

Παράδειγμα: Κλείσιμο Αρχείων

```c
#include <stdio.h>
int main() {
    FILE *fileToRead, *fileToWrite;
    fileToRead = fopen("input.txt", "r");
    ...
    fileToWrite = fopen("output.txt", "w");
    ...
    fclose(fileToRead);
    fclose(fileToWrite);
    return 0;
}
```

Σε κάθε fopen πρέπει να αντιστοιχεί ένα fclose

Η συνάρτηση `fread`

Η συνάρτηση `fread` μας επιτρέπει να διαβάσουμε bytes από ένα ανοιχτό αρχείο. Επιστρέφει το πλήθος των δεδομένων που διαβάστηκαν . Η δήλωση της συνάρτησης είναι

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *restrict stream);
```

ptr: δείκτης στην μνήμη όπου θα αποθηκευτούν τα δεδομένα εισόδου

size: ο αριθμός bytes κάθε δεδομένου που θα διαβαστεί

nmemb: πόσα δεδομένα να προσπαθήσει να διαβάσει

stream: δείκτης στο ανοιχτό αρχείο

Παράδειγμα: Διάβασμα Αρχείων

```c
#include <stdio.h>

int main() {

  FILE *fileToRead, *fileToWrite;

  fileToRead = fopen("input.txt", "r");

  if (!fileToRead) return 1;

  char buffer[1024];

  size_t bytesRead = fread(buffer, sizeof(char), 1023, fileToRead);

  buffer[bytesRead] = '\0';

  printf("# of bytes read: %d\n", bytesRead);

  printf("String read: %s\n", buffer);

  fclose(fileToRead);

  return 0;

}
```

Παράδειγμα: Διάβασμα Αρχείων

```c
#include <stdio.h>

int main() {

  FILE *fileToRead, *fileToWrite;

  fileToRead = fopen("input.txt", "r");

  if (!fileToRead) return 1;

  char buffer[1024];

  size_t bytesRead = fread(buffer, sizeof(char), 1023, fileToRead);

  buffer[bytesRead] = '\0';

  printf("# of bytes read: %d\n", bytesRead);

  printf("String read: %s\n", buffer);

  fclose(fileToRead);

  return 0;

}
```

```
$ ./fread
$ echo hello > input.txt
$ ./fread
# of bytes read: 6
String read: hello
```

Παράδειγμα #2: Διάβασμα Αρχείων

```c
#include <stdio.h>

int main() {

  FILE *fileToRead, *fileToWrite;

  fileToRead = fopen("input.txt", "r");

  if (!fileToRead) return 1;

  int buffer[1024];

  size_t integersRead = fread(buffer, sizeof(int), 1023, fileToRead);

  printf("# of integers read: %d\n", integersRead);

  printf("Integer read: %d %08x\n", buffer[0], buffer[0]);

  fclose(fileToRead);

  return 0;

}
```

Παράδειγμα #2: Διάβασμα Αρχείων

```c
#include <stdio.h>

int main() {

  FILE *fileToRead, *fileToWrite;

  fileToRead = fopen("input.txt", "r");

  if (!fileToRead) return 1;

  int buffer[1024];

  size_t integersRead = fread(buffer, sizeof(int), 1023, fileToRead);

  printf("# of integers read: %d\n", integersRead);

  printf("Integer read: %d %08x\n", buffer[0], buffer[0]);

  fclose(fileToRead);

  return 0;

}
```

```
$ ./freadint
# of integers read: 1
Integer read: 1819043176 6c6c6568
```

What?!

```
$ hexdump -C input.txt
00000000  68 65 6c 6c 6f 0a  |hello.|
```

Η συνάρτηση `fwrite`

Η συνάρτηση `fwrite` μας επιτρέπει να γράψουμε έναν αριθμό δεδομένων σε ένα αρχείο. Επιστρέφει τον αριθμό των στοιχείων που γράφτηκαν. Η δήλωση της συνάρτησης είναι

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *restrict stream);
```

ptr: δείκτης στην μνήμη απ'όπου θα διαβάσουμε τα δεδομένα εξόδου

size: ο αριθμός bytes κάθε δεδομένου που θα γραφτεί

nmemb: πόσα δεδομένα να προσπαθήσει να γράψει

stream: δείκτης στο ανοιχτό αρχείο

Παράδειγμα: Γράψιμο Αρχείων

```c
#include <stdio.h>
int main() {
  FILE *fileToWrite;
  fileToWrite = fopen("output.txt", "w");
  if (!fileToWrite) return 1;
  int numbers[4] = {0x42, 0x43, 0x44, 0x45};
  size_t numsWritten = fwrite(numbers, sizeof(int), 4, fileToWrite);
  printf("Wrote: %zu numbers\n", numsWritten);
  fclose(fileToWrite);
  return 0;
}
```

Παράδειγμα: Γράψιμο Αρχείων

```c
#include <stdio.h>

int main() {

  FILE *fileToWrite;

  fileToWrite = fopen("output.txt", "w");

  if (!fileToWrite) return 1;

  int numbers[4] = {0x42, 0x43, 0x44, 0x45};

  size_t numsWritten = fwrite(numbers, sizeof(int), 4, fileToWrite);

  printf("Wrote: %zu numbers\n", numsWritten);

  fclose(fileToWrite);

  return 0;

}
```

```
$ ./fwrite
Wrote: 4 numbers
$ hexdump -C output.txt
00000000  42 00 00 00 43 00 00 00  44 00 00 00 45 00 00 00
|B...C...D...E...|
```

File Descriptor (FD)

Κάθε ανοιχτό αρχείο για ένα πρόγραμμα έχει έναν μοναδικό αριθμό που λέγεται file descriptor. Μπορούμε να βρούμε αυτόν τον αριθμό με την συνάρτηση fileno.

```
FILE *fileToRead, *fileToWrite;

fileToRead = fopen("input.txt", "r");

fileToWrite = fopen("output.txt", "w");

// ...

printf("fileToRead: %d\n", fileno(fileToRead));

printf("fileToWrite: %d\n", fileno(fileToWrite));

printf("stdin: %d\n", fileno(stdin));

printf("stdout: %d\n", fileno(stdout));

printf("stderr: %d\n", fileno(stderr));
```

```
$ ./fileno
fileToRead: 3
fileToWrite: 4
stdin: 0
stdout: 1
stderr: 2
```

stdin, stdout και stderr

Τα stdin, stdout και stderr είναι δείκτες σε ανοικτά αρχεία που αρχικοποιούνται όταν το πρόγραμμα ξεκινά την εκτέλεσή του και κλείνουν όταν τερματίζει.

stdin: αντιστοιχεί στην πρότυπη είσοδο του προγράμματος και συνήθως έχει file descriptor 0.

stdout: αντιστοιχεί στην πρότυπη έξοδο του προγράμματος και συνήθως έχει file descriptor 1.

stderr: αντιστοιχεί στην έξοδο σφάλματος του προγράμματος και συνήθως έχει file descriptor 2.

Σύγκρινε το `find / -name foo` με το `find / -name foo 2> error.txt`

Η συνάρτηση `fscanf`

Η συνάρτηση `fscanf` είναι όμοια με την scanf, απλά αντί να διαβάζει από το stdin, μπορεί να διαβάσει από οποιοδήποτε αρχείο. Η συνάρτηση έχει την ακόλουθη μορφή:

```
int fscanf(FILE *stream, const char *format, ...);
```

Η κλήση scanf(x, y, z) είναι ουσιαστικά ισοδύναμη με την fscanf(stdin, x, y, z)

Η συνάρτηση `fprintf`

Η συνάρτηση `fprintf` είναι όμοια με την printf, απλά αντί να γράφει στο stdout, μπορεί να γράψει σε οποιοδήποτε αρχείο. Η συνάρτηση έχει την ακόλουθη μορφή:

```
int fprintf(FILE *stream, const char *format, ...);
```

Η κλήση printf(x, y, z) είναι ουσιαστικά ισοδύναμη με την fprintf(stdout, x, y, z)

Παράδειγμα χρήσης fprintf/fscanf

```c
#include <stdio.h>

int main() {

        FILE *fileToRead, *fileToWrite;

        fileToRead = fopen("input.txt", "r");

        fileToWrite = fopen("output.txt", "w");

        // ...

        int num;

        fscanf(fileToRead, "%d", &num);

        fprintf(fileToWrite, "Number: %d\n", num);

        // ...

        return 0;

}
```

Παράδειγμα χρήσης fprintf/fscanf

```c
#include <stdio.h>

int main() {

        FILE *fileToRead, *fileToWrite;

        fileToRead = fopen("input.txt", "r");

        fileToWrite = fopen("output.txt", "w");

        // ...

        int num;

        fscanf(fileToRead, "%d", &num);

        fprintf(fileToWrite, "Number: %d\n", num);

        // ...

        return 0;

}
```

```
$ echo "     42" > input.txt
$ ./stream
$ cat output.txt
Number: 42
```

Άλλες χρήσιμες συναρτήσεις

```
char *fgets(char *s, int size, FILE *stream);

int feof(FILE *stream);

int fseek(FILE *stream, long offset, int whence);

int fgetc(FILE *stream);

int fputc(int c, FILE *stream);
```

Για την επόμενη φορά

Από τις σημειώσεις καλύψαμε τις σελίδες 136-151, 160-177

- Visualizing sorting algorithms [1], [2], [3]
- Sorting algorithm (περιέχει εκτενή λίστα με όλους τους αλγορίθμους)
- Quicksort analysis
- Divide and conquer
- Scanf specifiers και άλλα
- Fread
- File descriptor
- Endianness

Ευχαριστώ και καλή μέρα εύχομαι!
Keep Coding ;)