

# Διάλεξη 14η: Διαχείριση Αρχείων

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών



- Στο `stdio.h`
  - Δηλώνει τον τύπο δεδομένων `FILE` με όλες τις πληροφορίες που χρειάζονται για την περιγραφή και προσπέλαση ενός αρχείου
    - Η δήλωση `FILE *f` δημιουργεί ένα δείκτη `f` σε `FILE`
  - Ορίζει 3 προκαθορισμένα αρχεία
    - `stdin` - η κύρια είσοδος
    - `stdout` - η κύρια έξοδος
    - `stderr` - η κύρια έξοδος για λάθη
  - Ορίζει τις τιμές των σταθερών
    - `NULL` - Δείκτης στο “πουθενά”
    - `EOF` - Το τέλος ενός αρχείου
  - Δηλώνει συναρτήσεις προσπέλασης και διαχείρισης αρχείων



- Η C βλέπει κάθε αρχείο σαν μια σειρά από bytes
  - Κάθε αρχείο “κλείνει” με end-of-file marker
- Μπορούμε να ανοίξουμε ένα αρχείο για διάβασμα ή για γράψιμο
- Όταν ανοίγει ένα αρχείο επιστρέφεται ένας δείκτης σε **FILE**



- Δήλωση

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

```
FILE *fopen(const char *fname, const char *mode);
```

- Παίρνει δύο ορίσματα

- Το `fname` είναι το όνομα του αρχείου που θα ανοιχτεί
- Το `mode` καθορίζει αν το αρχείο θα ανοιχτεί για ανάγνωση ή για εγγραφή

- Επιστρέφει έναν δείκτη σε `FILE` που αναπαριστά το ανοικτό αρχείο

- Σε περίπτωση αποτυχίας επιστρέφει `NULL`



# Τρόποι ανοίγματος αρχείου

Τρόπος			Περιγραφή	Αρχική θέση
r	rb		Ανοίγει προς ανάγνωση	Αρχή
w	wb		Ανοίγει προς εγγραφή, δημιουργείται αν δεν υπάρχει. Διαγράφει και αντικαθιστά τα προηγούμενα περιεχόμενα.	Αρχή
a	ab		Ανοίγει προς επέκταση, δημιουργείται αν δεν υπάρχει.	Τέλος
r+	rb+	r+b	Ανοίγει προς ανάγνωση και εγγραφή.	Αρχή
w+	wb+	w+b	Ανοίγει προς ανάγνωση και εγγραφή. Διαγράφει και αντικαθιστά τα προηγούμενα περιεχόμενα.	Αρχή
a+	ab+	a+b	Ανοίγει προς ανάγνωση και εγγραφή, ή επέκταση αν υπάρχει ήδη.	Τέλος



# Λάθη στο άνοιγμα αρχείων

- Αν συμβεί λάθος κατά το άνοιγμα, η `fopen` επιστρέφει `NULL`
- Πρέπει να γίνεται έλεγχος για λάθος
- Η πρόσβαση σε δείκτη που είναι `NULL` προκαλεί λάθος μνήμης (segmentation fault)

```
FILE *f;  
  
f = fopen("output_file", "w");  
if(f == (FILE *) NULL) {  
    fprintf(stderr, "Cannot open file\n");  
    exit(-1);  
}
```



# Σειριακή προσπέλαση

- Δημιουργία αρχείου κειμένου
- Αν υπάρχει ήδη αντικαθίστώνται τα περιεχόμενά του

file1.c

```
int main()
{
    FILE *fp;
    int index;

    fp = fopen("mytestfile.txt", "w");
    if(fp == (FILE *)NULL) {
        return -1;
    }
    for(index = 1; index <= 10; index++) {
        fprintf(fp, "Line number %d\n", index);
    }
    fclose(fp);
}
```



# Ανάγνωση και προβολή αρχείου κειμένου

- Ανάγνωση ενός αρχείου κειμένου
- Αν δεν υπάρχει το αρχείο επιστρέφει λάθος

cat.c

```
int main()
{
    FILE *fp;
    int c;

    fp = fopen("mytestfile.txt", "r");
    if(fp == (FILE *)NULL) {
        fprintf(stderr, "File doesn't exist\n");
        return -1;
    }
    do {
        c = getc(fp);
        putchar(c);
    } while (c != EOF);
    fclose(fp);
}
```





# Αντιγραφή αρχείου σε άλλο

- Ανάγνωση ενός αρχείου
- Εγγραφή όλων των περιεχομένων σε άλλο αρχείο

cp.c

```
void filecopy(FILE *inf, FILE *outf) {  
    int c;  
    while ((c = getc(inf)) != EOF) {  
        putc(c, outf);  
    }  
}
```



# Αντιγραφή αρχείου σε άλλο (2)

cp.c

```
int main(int argc, char **argv)
{
    FILE *input, *output;

    if(argc != 3) {
        fprintf(stderr, "usage: %s <from> <to>", argv[0]);
        return -1;
    }
    input = fopen(argv[1], "r");
    output = fopen(argv[2], "w");
    if( input == (FILE *)NULL
        || output == (FILE *)NULL)
    {
        fprintf(stderr, "Cannot open files\n");
        return -1;
    }
    filecopy(input, output);
    fclose(input);
    fclose(output);
}
```



# Επέκταση αρχείου

- Επεκτείνουμε ένα αρχείο γράφοντας στο τέλος του
- Το μέγεθος του αρχείου μεγαλώνει

append.c

```
#include <stdio.h>
int main()
{
    FILE *fp = fopen("file.txt", "a");
    fprintf(fp, "%s\n", "This is an extra line!");
    fclose(fp);
    return 0;
}
```



# Κλείσιμο αρχείων

- Ό,τι ανοίγει πρέπει να κλείνει
- Το πρόγραμμα δεν πρέπει να αφήνει ανοιχτά αρχεία
  - Μπορεί να μην εμφανιστούν όλες οι αλλαγές στο αρχείο
  - Σε διάφορα λειτουργικά συστήματα υπάρχει μέγιστος αριθμός ανοιχτών αρχείων
  - Δυσχεραίνει η πρόσβαση σε ανοιχτά αρχεία από άλλα προγράμματα
- Η συνάρτηση `fclose` κλείνει ένα αρχείο που άνοιξε η `fopen`

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

- Επιστρέφει 0 αν επιτύχει και EOF διαφορετικά
- Γίνεται αυτόματα στο τέλος του προγράμματος
  - Είναι καλή πρακτική να γίνεται από τον προγραμματιστή



# Ανάγνωση και εγγραφή ανά χαρακτήρα

- Συναρτήσεις που διαβάζουν και γράφουν χαρακτήρες

```
int fgetc(FILE *fstream);  
int fputc(int c, FILE *fstream);
```

- Η συνάρτηση `fgetc` διαβάζει ένα χαρακτήρα από το αρχείο
- Η συνάρτηση `fputc` γράφει ένα χαρακτήρα στο αρχείο



# Ανάγνωση και εγγραφή ανά γραμμή

- Συναρτήσεις που διαβάζουν και γράφουν γραμμές

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



# Μορφοποιημένη ανάγνωση και εγγραφή

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

- Αντίστοιχες των `printf` και `scanf`
- Επιπλέον το πρώτο όρισμα καθορίζει το αρχείο εισόδου ή εξόδου



# Τέλος αρχείου

- Οι συναρτήσεις ανάγνωσης συνήθως επιστρέφουν EOF όταν φτάσουν στο τέλος του αρχείου
- Η συνάρτηση `fEOF` ελέγχει αν φτάσαμε στο τέλος

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

- Επιστρέφει διαφορετικό του μηδέν ( $\neq 0$ ) για τέλος αρχείου, ή 0 αλλιώς

## Παράδειγμα

```
while (!feof(input_file)) {  
    if (fscanf(input_file, "%s %d", username, &score) != 2) {  
        break;  
    }  
    fprintf(output_file, "%s %d", username, score+10);  
}
```





# Προκαθορισμένα αρχεία

- Η βιβλιοθήκη `stdio.h` ορίζει τρία ειδικά αρχεία `FILE*`

```
FILE *stdin, *stdout, *stderr;
```

- Τα αρχεία `stdin`, `stdout`, `stderr` είναι αυτόματα ανοικτά σε κάθε C πρόγραμμα



## Προκαθορισμένα αρχεία (2)

- Το λειτουργικό σύστημα ανοίγει τα τρία ειδικά αρχεία πριν αρχίσει να εκτελεί το πρόγραμμα
  - Το ειδικό αρχείο `stdin` αντιστοιχεί στο πληκτρολόγιο-φαίνεται σαν ένα αρχείο που περιέχει ό,τι πληκτρολογείται
  - Το ειδικό αρχείο `stdout` αντιστοιχεί στην οθόνη
  - Το ειδικό αρχείο `stderr` αντιστοιχεί στην οθόνη (για μηνύματα λάθους)



# Ανακατεύθυνση στο UNIX

- Το λειτουργικό σύστημα UNIX (και το Linux) μπορεί να αντικαταστήσει τις προκαθορισμένες τιμές για τα `stdin`, `stdout`, `stderr` με οποιοδήποτε αρχείο
  - Εκτελώντας:  
`./a.out < inputfile`  
Το UNIX θέτει το αρχείο `stdin` να αντιστοιχεί στο `inputfile` αντί για το πληκτρολόγιο
  - Εκτελώντας:  
`./a.out > outputfile`  
Το UNIX θέτει το αρχείο `stdout` να αντιστοιχεί στο `outputfile` αντί για την οθόνη
  - Εκτελώντας:  
`./a.out >& errorfile`  
Το UNIX θέτει το αρχείο `stderr` να αντιστοιχεί στο `errorfile` αντί για την οθόνη
  - Μπορούν να συνδυαστούν:  
`./a.out < input.txt > output.txt >& error.txt`



# Τυχαία προσπέλαση

- Μία από τις πληροφορίες που διατηρούνται στις δομές τύπου `FILE` για κάθε αρχείο είναι η τρέχουσα θέση εγγραφής/ανάγνωσης
- Όταν κάνουμε εγγραφή κειμένου σε ένα αρχείο με τις συναρτήσεις `fputc`, `fputs`, `fprintf` η εγγραφή γίνεται στην τρέχουσα θέση μέσα στο αρχείο
- Όταν κάνουμε ανάγνωση κειμένου με τις συναρτήσεις `fgetc`, `fgets`, `fscanf` η ανάγνωση γίνεται από την τρέχουσα θέση στο αρχείο
- Η τρέχουσα θέση αυξάνει ανάλογα με τους χαρακτήρες που έχουμε γράψει ή διαβάσει
- Η τρέχουσα θέση όταν ανοίγει αρχικά ένα αρχείο εξαρτάται από το όρισμα `mode` της `fopen`
- Μπορούμε να αλλάξουμε την τρέχουσα θέση με την συνάρτηση `fseek`



```
int fseek(FILE *stream, long offset, int whence);  
long ftell(FILE *stream);  
void rewind(FILE *stream);
```

- Η `fseek` αλλάζει την τρέχουσα θέση στο αρχείο κατά `offset` bytes
  - `SEEK_SET` - σε σχέση με την αρχή
  - `SEEK_CUR` - σε σχέση με την τρέχουσα θέση
  - `SEEK_END` - σε σχέση με το τέλος
- Η `ftell` επιστρέφει την τρέχουσα θέση στο αρχείο
- Η `rewind` θέτει την τρέχουσα θέση στην αρχή του αρχείου



# Διαδικά αρχεία

- Είδαμε αρχεία κειμένου στα οποία γράφουμε χαρακτήρες
  - Πολλές φορές αποθηκεύουμε δεδομένα άλλης μορφής
- Διαδικό αρχείο: αρχείο που δεν περιέχει κείμενο αλλά πιθανώς και άλλες αναπαραστάσεις δεδομένων



# Ανάγνωση και εγγραφή

- Συναρτήσεις για την εγγραφή και ανάγνωση δυαδικών αρχείων

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

- Η `fread` διαβάζει από το αρχείο `stream` `nmemb` στοιχεία μεγέθους `size` το καθένα και τα αποθηκεύει στη μνήμη που δείχνει ο `ptr`
- Η `fwrite` γράφει στο αρχείο `stream` `nmemb` στοιχεία μεγέθους `size` το καθένα που βρίσκονται στη μνήμη που δείχνει ο `ptr`
- Ο δείκτης `ptr` δεν χρειάζεται να δείχνει σε δυναμική μνήμη (μπορεί να είναι και η διεύθυνση μιας απλής μεταβλητής `&x`)



# Ανάγνωση και εγγραφή (2)

- Η `fwrite` γράφει σε ένα δυαδικό αρχείο

## Παράδειγμα

```
fwrite(&number, sizeof(int), 1, out_file);
```

- `&number` - Δείκτης στη μνήμη που περιέχει τα δεδομένα
- `sizeof(int)` - Μέγεθος ενός στοιχείου από τα δεδομένα που υπάρχουν στην διεύθυνση `&number`
- `1` - Αριθμός των στοιχείων που υπάρχουν στη διεύθυνση `&number`
- `out_file` - Δείκτης στη δομή `FILE` που περιγράφει το αρχείο στο οποίο θα γίνει η εγγραφή

