

Εισαγωγή στο περιβάλλον προγραμματισμού του εργαστηρίου

Λειτουργικά Συστήματα

7ο εξάμηνο ΣΗΜΜΥ

ακ. έτος 2017-2018

<http://www.cslab.ece.ntua.gr/courses/os>



Εργαστήριο Υπολογιστικών Συστημάτων
ΕΜΠ

Οκτώβριος 2017

Εργαστήριο

- ▶ Εργαστηριακές Ασκήσεις
- ▶ Πέμπτη
 - ▶ 10:45-12:30
- ▶ Παρασκευή
 - ▶ 10:45-12:30
 - ▶ 12:45-14:30
- ▶ Εξέταση ασκήσεων: ασκήσεις που δεν εξετάζονται εμπρόθεσμα θα έχουν βαθμολογική επιβάρυνση

Ασκήσεις

- ▶ Περιβάλλον Linux
- ▶ Προγραμματισμός σε C

Τυπική διαδικασία:

- ▶ Εκφώνηση
- ▶ Παρουσίαση
(περιέχει ό,τι χρειάζεται για τη λύση)
- ▶ Επίδειξη
- ▶ Αναφορά (μία εβδομάδα μετά)
 - ▶ κώδικας
 - ▶ έξοδος προγράμματος
 - ▶ απάντησεις σε ερωτήσεις
 - ▶ Σύνοπτικές απαντήσεις!

Βιβλιογραφία για το εργαστήριο

Βιβλία

- ▶ The C Programming Language (K&R)
Brian Kernighan and Dennis Ritchie
- ▶ Advanced Programming in the UNIX® Environment
W. Richard Stevens
- ▶ The Linux Programming Interface
Michael Kerrisk
- ▶ Linux System Programming:
Talking Directly to the Kernel and C Library
Robert Love

Σύνδεσμοι τεκμηρίωσης

- ▶ Linux man-pages:
<http://www.kernel.org/doc/man-pages/>
- ▶ GNU C library:
<http://www.gnu.org/software/libc/manual/>

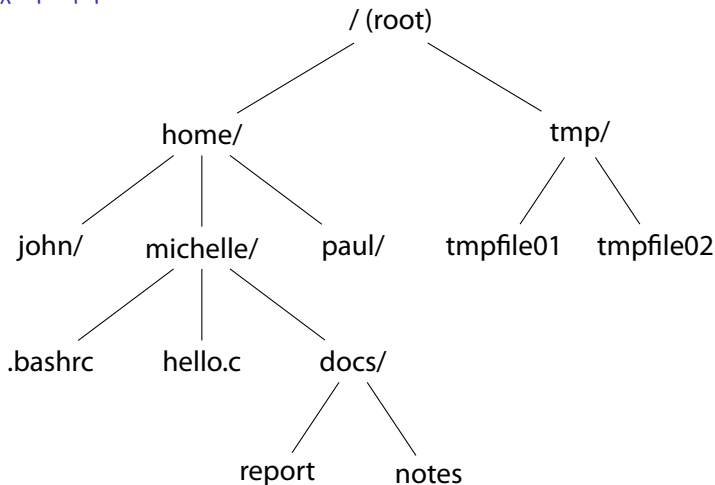
Διαδικαστικά

- ▶ Λογαριασμοί (Accounts)
 - ▶ Χρήστης: oslabXY
 - ▶ Αλλαγή password: yppasswd
- ▶ Μηχανήματα
 - ▶ orion.cslab.ece.ntua.gr
 - ▶ student-pc: Εκτέλεση από το orion, για σύνδεση σε τυχαίο μηχάνημα φοιτητών
- ▶ Λοιπά
 - ▶ Σύνδεση: ssh
 - ▶ Windows: putty, winscp
 - ▶ Linux: ssh/scp/sftp
 - ▶ **δεν** υποστηρίζονται USB sticks
 - ▶ Mailing list

Το περιβάλλον προγραμματισμού

Σύστημα αρχείων

Ιεραρχική δομή



- ▶ Κατάλογοι
- ▶ Αρχεία

Μονοπάτια στο σύστημα αρχείων

Μονοπάτι (paths):

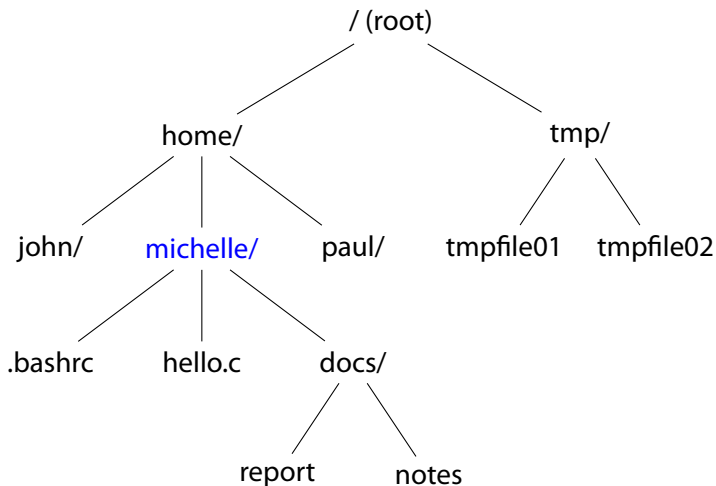
Συμβολοσειρα από αναγνωριστικά χωρισμένα από τον χαρακτήρα /
πχ: `/this/is/a/path/name`

Κανόνες:

- ▶ Το μονοπάτι είναι
 1. *απόλυτο* αν ξεκινάει με / -- αφετηρία είναι η αρχή της ιεραρχίας
 2. *σχετικό* (αν όχι) -- αφετηρία είναι ο τρέχων κατάλογος (TK)
- ▶ Το αναγνωριστικό:
 - . σηματοδοτεί τον TK
 - .. σηματοδοτεί τον πατέρα του TK

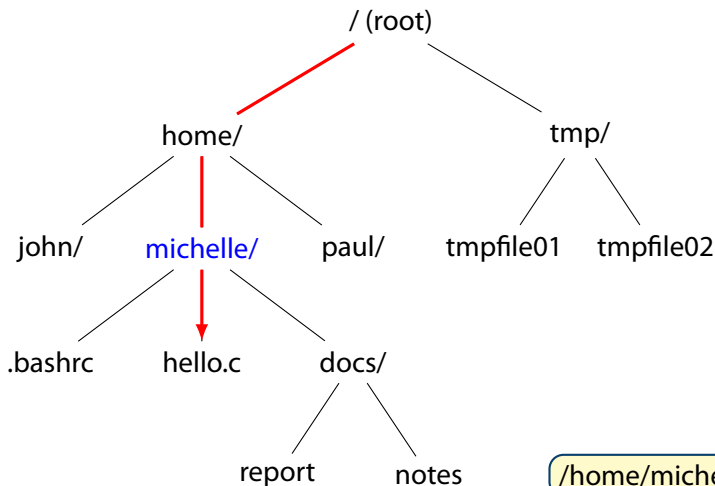
Παραδείγματα μονοπατιών

TK: /home/michelle



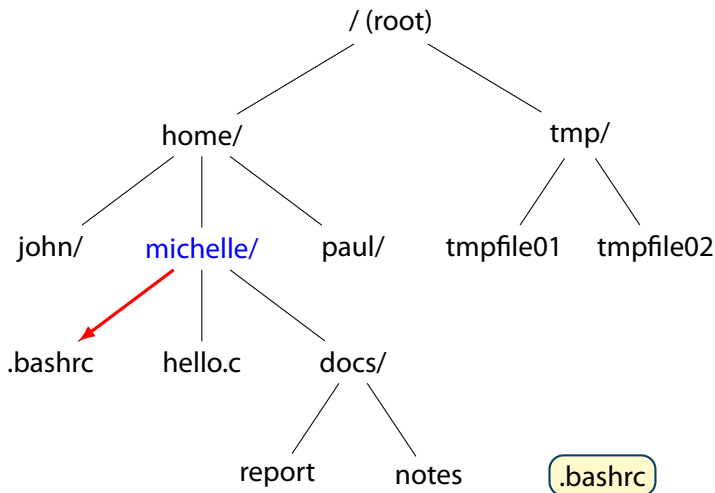
Παραδείγματα μονοπατιών

TK: /home/michelle



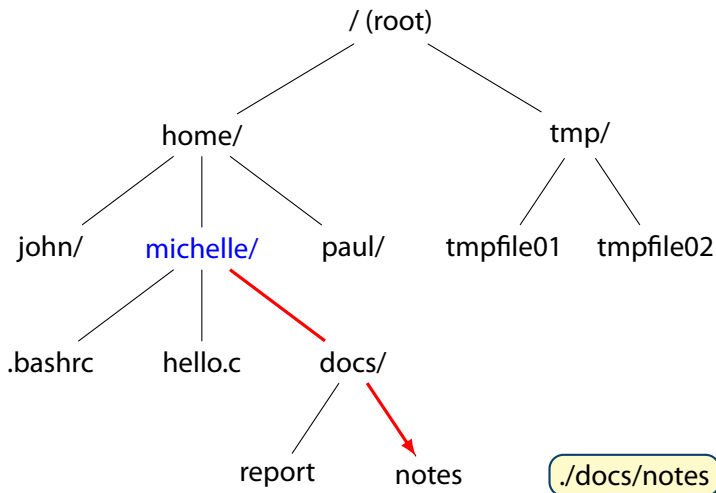
Παραδείγματα μονοπατιών

TK: /home/michelle



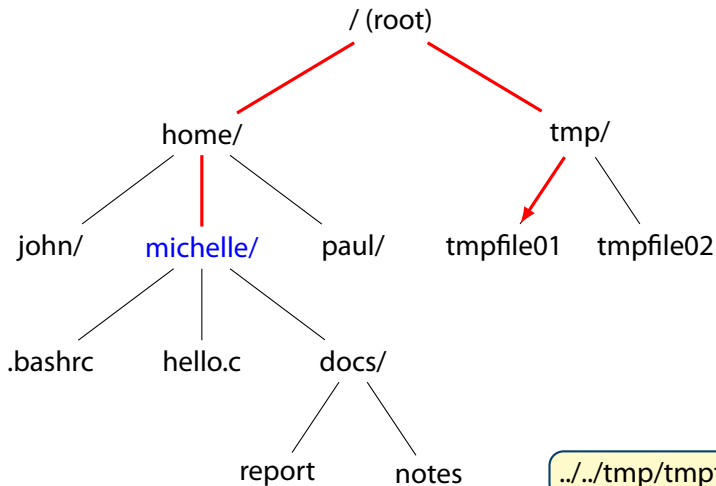
Παραδείγματα μονοπατιών

TK: /home/michelle



Παραδείγματα μονοπατιών

TK: /home/michelle



Ο φλοιός (shell)

- ▶ prompt (↑,↓,<TAB>)
- ▶ Προσωπικός κατάλογος (home directory): \$HOME,~
- ▶ Τρέχων κατάλογος (current directory)

Γραμμή εντολών

```
$ echo $HOME
/home/oslab/oslabf23
$ pwd
/home/oslab/oslabf23
$ cd /home/
$ pwd
/home
$ cd ~
$ pwd
/home/oslab/oslabf23
```

Διαχείριση καταλόγων

Εντολές

- `cd`: Αλλαγή τρέχοντος καταλόγου
- `mkdir`: Δημιουργία καταλόγου
- `rmdir`: Διαγραφή καταλόγου

Γραμμή εντολών

```
$ mkdir dir  
$ cd dir/  
$ mkdir -p 1/2/3/4  
$ ls 1/2/3  
4  
$ rmdir -p 1/2/3/4  
$ cd ../  
$ rmdir dir/
```

Διαχείριση Αρχείων (+Ανακατεύθυνση)

Εντολές

- `cat`: Εκτύπωση
- `cp`: Αντιγραφή
- `mv`: Μετακίνηση
- `rm`: Διαγραφή

Γραμμή εντολών

```
$ echo "Hello World" > file
$ cat file
Hello World
$ mv file file-1
$ cp file-1 file-2
$ cat file-1 file-2
Hello World
Hello World
$ rm file-1 file-2
```


Τεκμηρίωση (Documentation)

- ▶ Η εντολή `man` (`man man`)
- ▶ Τμήματα (sections) (`man 3 printf -- printf(3)`)
- ▶ `man -k` (`man -k printf`)

Γραμμή εντολών

```
$ man 2 read
```

NAME

```
read - read from a file descriptor
```

SYNOPSIS

```
#include <unistd.h>
```

```
ssize_t read(int fd, void *buf, size_t count);
```

DESCRIPTION

```
read() attempts to read up to count bytes from  
file descriptor fd into the buffer starting at buf.
```

Editor(s)

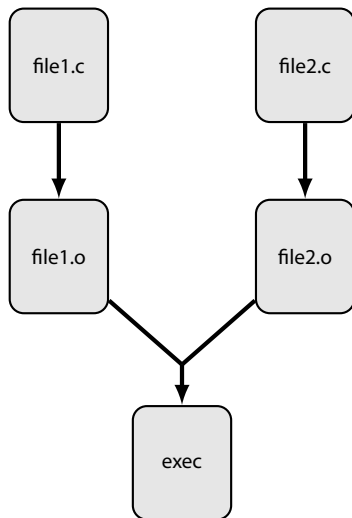
- ▶ vim
 - ▶ Δύο καταστάσεις:
 - ▶ εντολών
 - ▶ εισαγωγής κειμένου
 - ▶ gvim
- ▶ nano
- ▶ gedit
- ▶ ...

Μεταβλητές περιβάλλοντος

- ▶ \$HOME: Προσωπικός κατάλογος χρήστη
- ▶ \$PATH: Λίστα καταλόγων που περιέχουν εκτελέσιμα αρχεία.
Για προγράμματα που δεν υπάρχουν στο \$PATH, απαιτείται μονοπάτι
π.χ. ./myprog
- ▶ env(1), getenv(3)/setenv(3)

Παραγωγή εκτελέσιμου

1. Compile (Μεταγλώττιση):
 $\text{file1.c} \Rightarrow \text{file1.s} \Rightarrow \text{file1.o}$
2. Link:
 $\text{file1.o} + \text{file2.o} \Rightarrow \text{exec}$



Παράδειγμα: Hello World!

hello.c

```
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hello World!\n");
    return 0;
}
```

Γραμμή εντολών

```
$ gcc -Wall hello.c -o hello
$ ./hello
Hello World!
```

Παράδειγμα: Hello World! (compiling & linking)

Γραμμή εντολών

```
$ gcc -Wall -c hello.c  
$ gcc hello.o -o hello  
$ ./hello  
Hello World!
```

Πολλαπλά αρχεία

hello.c

```
#include <stdio.h>

void hello(const char *name)
{
    printf("Hello %s!\n", name);
}
```

main.c

```
void hello(const char *);

int
main(int argc, char **argv)
{
    hello("World");
    return 0;
}
```

Γραμμή εντολών

```
$ gcc -Wall -c main.c
$ gcc -Wall -c hello.c
$ gcc main.o hello.o -o hello
$ ./hello
Hello World!
```

Επικεφαλίδες

- ▶ Διεπαφή προς άλλα κομμάτια κώδικα (API)
- ▶ Περιέχουν πρότυπα και δηλώσεις
 - ▶ Συναρτήσεις
 - ▶ Καθολικές (global) μεταβλητές
- ▶ .h αρχεία
- ▶ preprocessor:
`#include "header.h"`

Επικεφαλίδες: Παράδειγμα

hello.h

```
void hello(const char *);
```

hello.c

```
#include <stdio.h>

void hello(const char *name)
{
    printf("Hello %s!\n", name);
}
```

main.c

```
#include "hello.h"

int main(int argc, char **argv)
{
    hello("World");
    return 0;
}
```

Ορίσματα Προγράμματος

- ▶ `int main(int argc, char **argv)`
 - `argc`: Αριθμός ορισμάτων προγράμματος
 - `argv`: Πίνακας με τα ορίσματα
 - `argv[0]`: Το όνομα του προγράμματος
- ▶ βοηθητική βιβλιοθήκη: `getopt(3)`

args.c

```
#include <stdio.h>
int main(int argc, char **argv)
{
    int i;
    for (i=0; i<argc; i++)
        printf("%d %s\n", i, argv[i]);
    return 0;
}
```

Γραμμή εντολών

```
$ ./args arg1 arg2
0 ./args
1 arg1
2 arg2
```

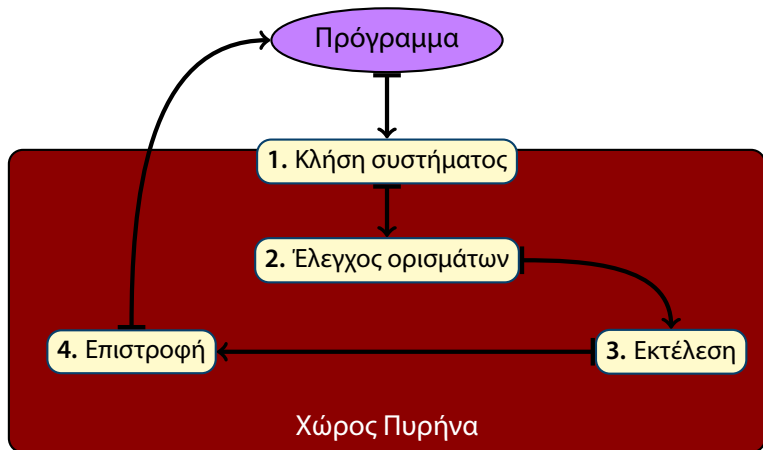
Αρχεία

- ▶ όνομα/μονοπάτι
(π.χ. /home/michelle/hello.c)
- ▶ δεδομένα/περιεχόμενα
(π.χ. `int main() ...`)
- ▶ μέτα-δεδομένα
(π.χ. ημερομηνία τελευταίας πρόσβασης)
- ▶ persistent
(παραμένουν μετά το κλείσιμο του υπολογιστή)
- ▶ "Everything is a file"
(ρητό του Unix)
- ▶ Για την πρόσβαση σε αυτά χρησιμοποιούμε:
κλήσεις συστήματος (system calls)

Κλήσεις συστήματος

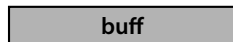
(system calls)

Προγραμματιστική διεπαφή για τις υπηρεσίες που προσφέρει το ΛΣ στις εφαρμογές.



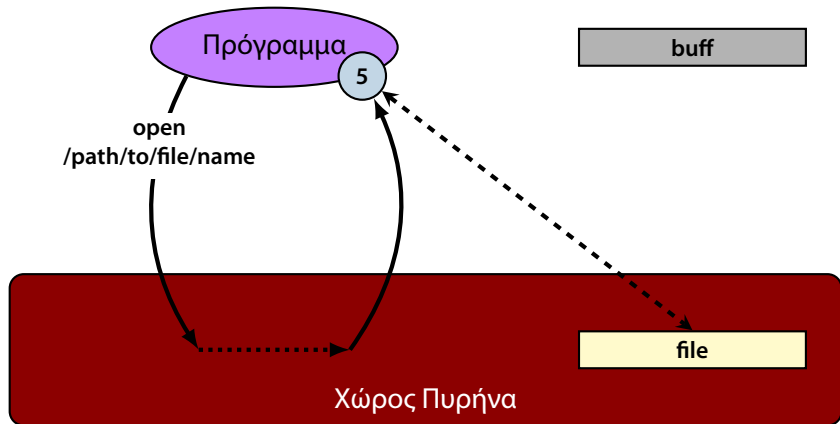
Βασικές κλήσεις Ε/Ε

(open,read,close)



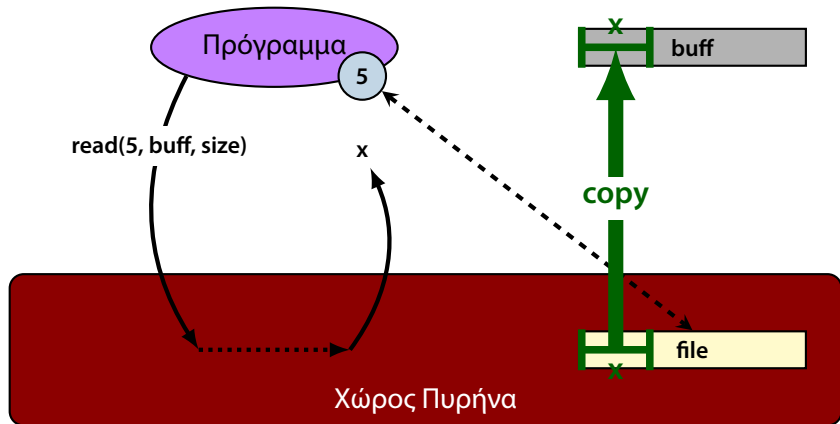
Βασικές κλήσεις E/E

(open,read,close)



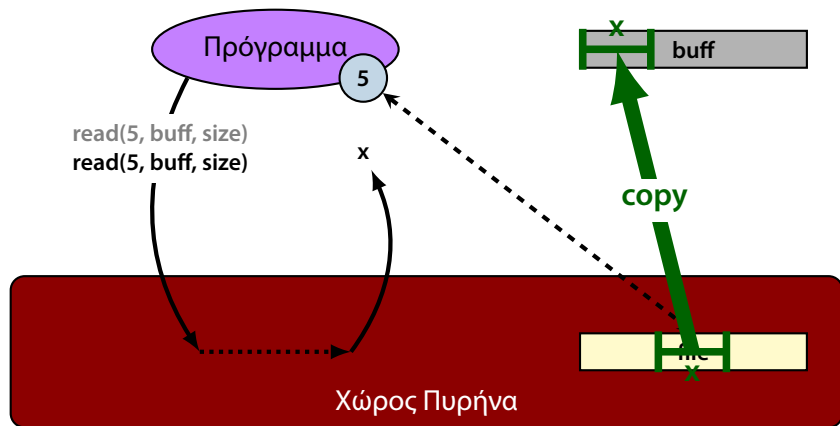
Βασικές κλήσεις E/E

(open,read,close)



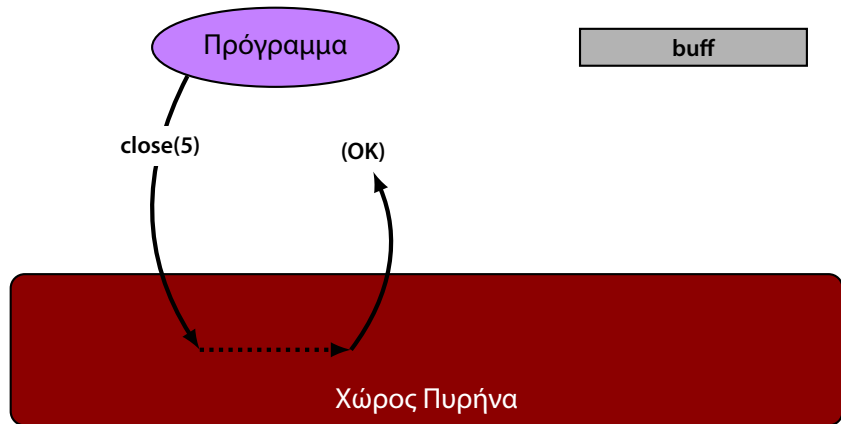
Βασικές κλήσεις E/E

(open,read,close)



Βασικές κλήσεις E/E

(open,read,close)



Βασικές κλήσεις Ε/Ε

Ανάγνωση:

- Άνοιγμα (open)
- Ανάγνωση σε μνήμη (read)
- Κλείσιμο (close)

Εγγραφή:

- Άνοιγμα (open)
- Εγγραφή από μνήμη (write)
- Κλείσιμο (close)

Από (εγγραφή) και Προς (ανάγνωση) τη μνήμη

υπάρχουν εξαιρέσεις (πχ `sendfile()`)

Η `open` επιστρέφει έναν ακέραιο (*file descriptor*) που λειτουργεί ως αναγνωριστικό για τις υπόλοιπες κλήσεις συστήματος (`read`, `write`, κλπ)

Διαθέσιμα αναγνωριστικά στην έναρξη του προγράμματος:

0: είσοδος (`stdin`)

1: έξοδος (`stdout`)

2: έξοδος σφαλμάτων (`stderr`)

Παράδειγμα

read

read.c

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

int main(int argc, char **argv)
{
    char buff[1024];
    ssize_t rcnt;
    for (;;) {
        rcnt = read(0, buff, sizeof(buff)-1);
        if (rcnt == 0) /* end-of-file */
            return 0;
        if (rcnt == -1) { /* error */
            perror("read");
            return 1;
        }
        buff[rcnt] = '\0';
        fprintf(stdout, "%s", buff);
    }
}
```

Παράδειγμα

write

write.c

```
#include <unistd.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv){
    char buff[1024];
    size_t len, idx;
    ssize_t wcnt;
    for (;;){
        if (fgets(buff,sizeof(buff),stdin) == NULL)
            return 0;
        idx = 0;
        len = strlen(buff);
        do {
            wcnt = write(1,buff + idx, len - idx);
            if (wcnt == -1){ /* error */
                perror("write");
                return 1;
            }
            idx += wcnt;
        } while (idx < len);
    }
}
```

Παράδειγμα

open (για ανάγνωση)

open-read.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    int fd;
    fd = open("filename", O_RDONLY);
    if (fd == -1){
        perror("open");
        exit(1);
    }
    // perform read(...)
    close(fd);
    return 0;
}
```

Παράδειγμα

open (για εγγραφή)

open-write.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    int fd, oflags, mode;
    oflags = O_CREAT | O_WRONLY | O_TRUNC;
    mode = S_IRUSR | S_IWUSR;
    fd = open("filename", oflags, mode);
    if (fd == -1){
        perror("open");
        exit(1);
    }
    // perform write(...)
    close(fd);
    return 0;
}
```

Παράδειγμα

open (για εγγραφή)

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int fd, oflags, mode;

    oflags = O_CREAT | O_WRONLY | O_TRUNC;
    mode = S_IRUSR | S_IWUSR;
    fd = open("filename", oflags, mode);
    if (fd == -1){
        perror("open");
        exit(1);
    }
    // perform write(...)
    close(fd);
    return 0;
}
```

Makefile(s)

- ▶ Εφαρμογή `make`
- ▶ Χρήση για την αυτόματη δημιουργία προγραμμάτων από αρχεία κώδικα.
- ▶ Διαδικασία:
 - ▶ Συγγραφή αρχείου `Makefile` που περιέχει κανόνες
 - ▶ Κλήση εντολής `make` για την παραγωγή του προγράμματος
- ▶ Κανόνες:
 - ▶ Αρχείο-στόχος (target)
 - ▶ Αρχεία-απαιτήσεις (prerequisites)
 - ▶ Εντολή παραγωγής στόχου από απαιτούμενα αρχεία

Παράδειγμα:

```
target : prerequisites  
    command
```


Παράδειγμα Makefile

Makefile

```
hello: hello.o main.o
    gcc -o hello hello.o main.o

hello.o: hello.c
    gcc -Wall -c hello.c

main.o: main.c
    gcc -Wall -c main.c
```

Γραμμή εντολών

```
$ make
gcc -Wall -c hello.c
gcc -Wall -c main.c
gcc -o hello hello.o main.o
$ make
make: `hello' is up to date.
$ edit main.c
$ make
gcc -Wall -c main.c
gcc -o hello hello.o main.o
```

strace

- ▶ Εφαρμογή
- ▶ Εκτέλεση προγράμματος που δίδεται ως όρισμα
- ▶ Καταγραφή των κλήσεων συστήματος που πραγματοποιούνται
- ▶ Χρήσιμο για εντοπισμό λαθών

Παράδειγμα strace

Γραμμή εντολών

```
$ echo 'Hello World!' > hello
$ cat hello
Hello World!
$ strace cat hello
execve("/bin/cat", ["cat", "hello"], [/* 52 vars */]) = 0
...
open("hello", O_RDONLY) = 3
read(3, "Hello World!\n", 32768) = 13
write(1, "Hello World!\n", 13Hello World!
) = 13
read(3, "", 32768) = 0
...
```

Στοίχιση

(indentation)

- ▶ Δεν είναι απαραίτητη για να μεταγλωττιστεί το πρόγραμμα
- ▶ Είναι απαραίτητη για να είναι κατανοητό

main

```
int main(void)
{
    blah();
    return 0;
}
```

if-else

```
void func()
{
    if (condition) {
        this();
        that();
    } else {
        other();
        whatever();
    }
}
```

- ▶ Πρόγραμμα cslab-indent
- ▶ Περισσότερες πληροφορίες:
<http://www.cslab.ece.ntua.gr/courses/os/ql/indent.html>

Στοίβα (stack)

- ▶ Αυτόματη αύξηση μεγέθους
- ▶ Όχι για πολλά δεδομένα (8 MB)

stack.c

```
void foo(double *);

int main(int argc, char **argv)
{
    double matrix[1048576];
    foo(matrix);
    return 0;
}
```

stack.s

```
sub    $0x8000008,%rsp
mov     %rsp,%rdi
callq   f <main+0xf>
xor     %eax,%eax
add     $0x8000008,%rsp
retq
```

Διαχείριση μνήμης

Σωρός (Heap) -- malloc / free

malloc.c

```
#include <stdio.h>
#include <stdlib.h>
void foo(double *);

int main(int argc, char **argv)
{
    double *array;
    array = malloc(1048576*sizeof(double));
    if (!array){
        printf("error in allocation\n");
        return 1;
    }
    foo(array);
    free(array);
    return 0;
}
```