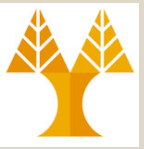
Επεξεργασία Δυαδικών Αρχείων

Στην ενότητα αυτή θα μελετηθούν:

Δυαδικά αρχεία

Συναρτήσεις Επεξεργασίας Δυαδικών Αρχείων

Αρχεία Κειμένου



- Θυμηθείτε ότι το **αρχείο** είναι μία ακολουθία bytes.
  π.χ. 01100011 01100001 01110010…

  ascii:99 ascii:97 ascii:114

  c a r
- Οι χαρακτήρες αυτοί είναι αποθηκευμένοι σειριακά στο αρχείο και διαχωρίζονται με διάφορους ειδικούς χαρακτήρες

Αρχείο όπως το βλέπει ο χρήστης

```
Car
Test    hello
```

Αρχείο στην Πραγματικότητα

Car\nTest\thelloEOF

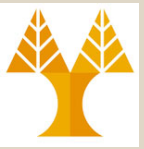
Τέλος αρχείου

Αρχεία Κειμένου

Ο Πίνακας ASCII – Αρκετοί χαρακτήρες είναι κρυμμένοι – δηλαδή δεν φαίνονται στην οθόνη και στα αρχεία

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Spa	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DE

Αρχεία Κειμένου και Δυαδικά Αρχεία



```
#include <stdio.h>

int main() {
        char c = 'a';
        int i = 10;
        FILE *bin, *text;

        bin = fopen("binfile.bin", "wb");
        text = fopen("textfile.txt", "w");

        fwrite(&i, sizeof (i), 1, bin);
        fwrite(&c, sizeof (c), 1, bin);
        fprintf(text, "%d", i);
        fprintf(text, "%c", c);
        fclose(bin);
        fclose(text);
        return 0;
}
```

Μέχρι τώρα μπορούσαμε να αποθηκεύσουμε δεδομένα σε αρχεία υπό μορφή κειμένου (δηλαδή ως ένα σύνολο χαρακτήρων).

textfile.txt
Contents: 10a
Size: 3 bytes
binfile.bin
Contents: μη αναγνώσιμα με text editor
Size: 5 bytes

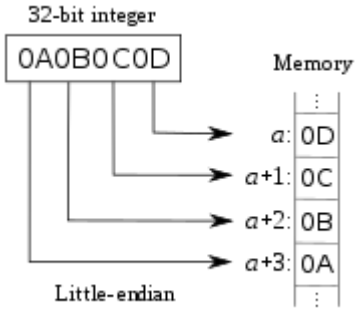
hexdump

- Το hexdump είναι ένα φίλτρο το οποίο εμφανίζει τα bytes των αρχείων σε μια καθορισμένη από τον χρήστη μορφή.
  - C : Canonical mode. Display hexadecimal offset, two sets of eight columns of hexadecimal bytes, then a | followed by the ASCII representation of those same bytes.

- hexdump -C binfile.bin

```
00000000  0a 00 00 00 61                                    |....a|
00000005
```

Στις αρχιτεκτονικές επεξεργαστών intel x86 τα δεδομένα αποθηκεύονται στη μνήμη ξεκινώντας από το least significant byte (little endian) =>
0a 00 00 00 είναι ο 32 bit αριθμός
00 00 00 0a = 10



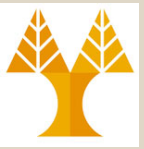
Τα πρώτα 4 bytes δεν μπορούν να απεικονιστούν σαν κάποιος χαρακτήρας ASCII, ενώ το 5ο είναι ο χαρακτήρας a.

hexdump

- hexdump -C ucy.gif

```
00000000  47 49 46 38 39 61 2c 01  12 01 70 00 00 21 f9 04  |GIF89a,...p..!..|
00000010  01 00 00 fc 00 2c 00 00  00 00 2c 01 12 01 87 00  |.....,....,.....|
00000020  00 00 00 00 33 00 00 66  00 00 99 00 00 cc 00 00  |....3..f........|
00000030  ff 00 2b 00 00 2b 33 00  2b 66 00 2b 99 00 2b cc  |..+..+3.+f.+..+.|
00000040  00 2b ff 00 55 00 00 55  33 00 55 66 00 55 99 00  |.+..U..U3.Uf.U..|
00000050  55 cc 00 55 ff 00 80 00  00 80 33 00 80 66 00 80  |U..U......3..f..|
00000060  99 00 80 cc 00 80 ff 00  aa 00 00 aa 33 00 aa 66  |............3..f|
00000070  00 aa 99 00 aa cc 00 aa  ff 00 d5 00 00 d5 33 00  |..............3.|
00000080  d5 66 00 d5 99 00 d5 cc  00 d5 ff 00 ff 00 00 ff  |.f..............|
00000090  33 00 ff 66 00 ff 99 00  ff cc 00 ff ff 33 00 00  |3..f.........3..|
000000a0  33 00 33 33 00 66 33 00  99 33 00 cc 33 00 ff 33  |3.33.f3..3..3..3|
000000b0  2b 00 33 2b 33 33 2b 66  33 2b 99 33 2b cc 33 2b  |+.3+33+f3+.3+.3+|
000000c0  ff 33 55 00 33 55 33 33  55 66 33 55 99 33 55 cc  |.3U.3U33Uf3U.3U.|
000000d0  33 55 ff 33 80 00 33 80  33 33 80 66 33 80 99 33  |3U.3..3.33.f3..3|
000000e0  80 cc 33 80 ff 33 aa 00  33 aa 33 33 aa 66 33 aa  |..3..3..3.33.f3.|
```

Δυαδικά Αρχεία και Δομές (structs)



- Όταν υπάρχει ανάγκη σε ένα πρόγραμμα να αποθηκεύσουμε ένα σύνολο πληροφοριών οι οποίες είναι οργανωμένες σε δομές (structures) σε ένα αρχείο κειμένου τότε θα πρέπει να ξεχωρίσουμε τα επιμέρους πεδία των δομών και να φυλάξουμε το κάθε ένα από αυτά σε μορφή κειμένου.

- Σε μεταγενέστερο στάδιο, όταν θέλουμε να επαναφέρουμε τα στοιχεία αυτά από το αρχείο στο πρόγραμμά μας, θα πρέπει να διαβάζουμε τα πεδία του αρχείου και να τα επαναφέρουμε ένα ένα στα πεδία των δομών.

- Αυτή η επίμονη διαδικασία, μπορεί να αποφευχθεί με την χρήση *δυαδικών αρχείων* (binary files). Αυτά τα αρχεία δεν είναι ούτε αναγνώσιμα ούτε εκτυπώσιμα. Μπορούμε να τα επεξεργαστούμε μόνο μέσω των προγραμμάτων μας.

Δυαδικά Αρχεία και Δομές (structs)

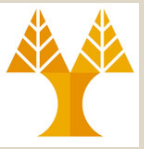
- Για άνοιγμα ενός δυαδικού αρχείου χρησιμοποιούμε όπως και για άνοιγμα αρχείων κειμένου τη συνάρτηση **`fopen`**, δίνοντας στο δεύτερο όρισμα της **`fopen`** το γράμμα **`b`** το οποίο ορίζει ότι θα πρέπει να ανοιχθεί δυαδικό αρχείο.

- Στο λειτουργικό σύστημα Unix η χρήση του **`b`** δεν είναι απαραίτητη, σε άλλα όμως συστήματα επιβάλλεται. Προτείνεται πάντα να χρησιμοποιείται.

- Για παράδειγμα η πιο κάτω δήλωση και εντολή έχουν σαν αποτέλεσμα το άνοιγμα του αρχείου “file.bin” ως δυαδικό αρχείο για ανάγνωση:

```
FILE  *fp;
fopen("file.bin", "rb");
```

Συναρτήσεις Επεξεργασίας Δυαδικών Αρχείων



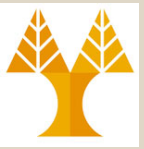
1. `int fwrite(*p, int s, int n, FILE* fp)`

   Η συνάρτηση αυτή γράφει στο αρχείο στο οποίο αναφέρεται ο δείκτης fp, στοιχεία πλήθους n, και μεγέθους s, η αρχή των οποίων καθορίζεται από τον δείκτη p. Επιστρέφεται το πλήθος των στοιχείων που γράφηκαν.

2. `int fread(*p, int s, int n, FILE* fp)`

   Η συνάρτηση αυτή διαβάζει από το αρχείο στο οποίο αναφέρεται ο δείκτης fp, στοιχεία πλήθους n, και μεγέθους s, και τα τοποθετεί στη μνήμη ξεκινώντας από εκεί που δείχνει ο δείκτης p. Επιστρέφεται το πλήθος των στοιχείων που διαβάστηκαν.

Παράδειγμα



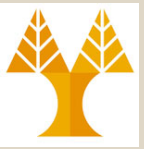
- Το πιο κάτω πρόγραμμα, γράφει στο αρχείο file.bin τις δέκα πρώτες δομές τύπου Person ενός πίνακα A και στην συνέχεια διαβάζει αυτές τις δέκα εγγραφές και τις τοποθετεί στον πίνακα B.

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

struct Person {
    char name[20];
    int age;
} A[20], B[10];

int main(void){
FILE *fp;
…
fp = fopen("file.bin", "wb");
fwrite(A, sizeof(struct Person), 10, fp);
fclose(fp);
```

Παράδειγμα



```
…

fp = fopen("file.bin", "rb");
fread(B, sizeof(struct Person), 10, fp);
fclose (fp);

…
```

Παράδειγμα (fread, fwrite)

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

#define NB 20
#define outFile "out.dat"

int main(void) {
  FILE *f_in, *f_out;
  int *tab1, *tab2;
  int i;

  tab1 = (int*)malloc(NB * sizeof(int));
  tab2 = (int*)malloc(NB * sizeof(int));

  for (i = 0 ; i < NB; i++)
    tab1[i] = i;

  /* writting to outFile*/
  if ((f_out = fopen(outFile, "w")) == NULL) {
    fprintf(stderr, "\nCan't write in file %s\n",outFile);
    return(EXIT_FAILURE);
  }
  fwrite(tab1, NB * sizeof(int), 1, f_out);
  fclose(f_out);
```

```
 /* reading from outFile */
  if ((f_in = fopen(outFile, "r")) == NULL) {
    fprintf(stderr, "\nCan't read from file %s\n",outFile);
    return(EXIT_FAILURE);
  }

  fread(tab2, NB * sizeof(int), 1, f_in);
  fclose(f_in);

  for (i = 0 ; i < NB; i++)
    printf("%d\t",tab2[i]);
  printf("\n");

  return(EXIT_SUCCESS);
}
```