Introdução à Ciência da Computação II

Revisão de Linguagem C
Pt. II: Arquivos

Prof. Ricardo J. G. B. Campello

Agradecimentos

- Parte dos slides a seguir são adaptações dos originais gentilmente cedidos por:
  - Prof. Rudinei Goularte
  - Profa. Maria Cristina F. de Oliveira

Sumário

- Noções Básicas de Arquivos
  - Arquivos Físico e Lógico
  - Arquivos Texto
  - Arquivos Binários
- Manipulação de Arquivos em C

Arquivos

- O que são?
  - São estruturas para armazenamento externo de dados
- Onde ficam os dados, nesse caso?
  - Em um dispositivo de memória secundária, como HDs, CDs, DVDs, memórias flash e fitas magnéticas
- Porque são necessários?
  - Quando o volume de dados é maior que o espaço disponível na memória principal
  - Quando se deseja armazenar informações de modo permanente (persistente, não volátil)

Arquivos

- Podem ser de dois tipos:
  - Arquivos do tipo texto
  - Arquivos binários
- Arquivos de texto
  - Armazena todas as informações, numéricas ou literais, através do código ASCII dos caracteres
  - Cada byte é visto como um caractere
- Arquivos binários
  - Significado dos bytes não é universal
  - Estrutura binária depende do programa que gerou o arquivo

Arquivo Físico e Arquivo Lógico

- **Arquivo Físico:**
  - seqüência de bytes armazenados no dispositivo
    - armazenamento em geral não é fisicamente seqüencial
- **Arquivo Lógico**:
  - arquivo como visto pelo aplicativo que o acessa
    - freqüentemente visão é seqüencial
- **Associação Arquivos Físico – Lógico**:
  - iniciada pelo aplicativo, gerenciada pelo S.O.

6

Arquivo Físico e Arquivo Lógico

- **Arquivo Físico**:
  - conjunto de bytes (p. ex. em disco)
    - geralmente agrupados em setores e clusters de dados
    - gerenciado pelo sistema operacional
- **Arquivo Lógico**:
  - modo como a *linguagem de programação* ou *aplicativo* enxerga os dados
    - seqüência de bytes, eventualmente organizados em *registros* ou outra *estrutura lógica*

7

Arquivos Texto

- Todas as informações, numéricas ou literais, são armazenadas através do código ASCII de seus caracteres.
- Exemplo:
  - Valor numérico 65 $\Rightarrow$ '6' e '5' $\Rightarrow$ 54 e 53 em ASCII $\Rightarrow$ bytes 00110110 e 00110101

Arquivos Texto – Tabela ASCII

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		Space	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;	DEL

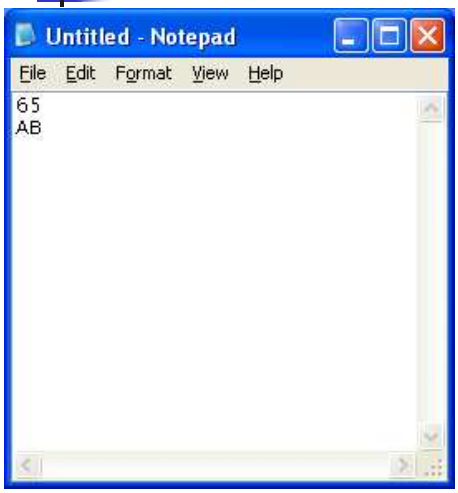
Source: www.asciitable.com

Arquivos Texto
Tabela ASCII Estendida

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	169	_	185	╣	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	▉	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	_
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	▒	192	└	208	╨	224	α	240	≡		

Source: www.asciitable.com

Arquivos Texto



0	0	1	1	0	1	1	0	54 = 6 (ASCII)
0	0	1	1	0	1	0	1	53 = 5 (ASCII)
0	0	0	0	1	1	0	1	13 = CR (ASCII)
0	0	0	0	1	0	1	0	10 = LF (ASCII)
0	1	0	0	0	0	0	1	65 = A (ASCII)
0	1	0	0	0	0	1	0	66 = B (ASCII)
0	0	0	0	1	1	0	1	13 = CR (ASCII)
0	0	0	0	1	0	1	0	10 = LF (ASCII)

- processador / editor de texto faz a conversão

Arquivos Texto

- Vantagens e Desvantagens:
  - Mais simples de compreender
    - e, em muitos casos, mais simples também de lidar
  - Conversão binário – ASCII é transparente e padrão
  - Permitem manipulação direta via editores simples
  - Restritos a lidar com unidades elementares (bytes)
    - o que é inapropriado em muitos casos
  - Arquivos texto contendo apenas infos. numéricas são maiores que os binários correspondentes

Arquivos Binários

- Significado dos bytes não é universal
- Estrutura depende do programa que gerou o arquivo
  - Dois Exemplos Muito Simples:
    - Arquivo apenas com números inteiros de 4 bytes
      - O 1º inteiro (4 bytes) indica a quantidade de números no arquivo
      - Um novo número a cada 4 bytes a partir do 5º byte
    - Arquivo com seqüência de registros do mesmo tipo
      - 2 campos string (10 + 30 bytes) e 2 inteiros (4 + 4 bytes)
      - Novo registro a cada 48 bytes até final do arquivo

Arquivos Binários

- Principais vantagens e desvantagens:
  - Permitem leitura/escrita (L/E) de variáveis da forma que são representadas em RAM
    - Permitem manipular variáveis compostas
    - Por exemplo, registros (structs)
  - São mais complexos que arquivos texto
    - É necessário conhecer a organização do arquivo

Associação Arquivo Físico – Lógico

- Em C: (associa e abre – nesse exemplo para escrita)

```
FILE *pt_arq;
if ( (pt_arq=fopen("meu_arq.txt", "w")) == NULL )
    printf("erro...")    /* p. ex. disco cheio ou protegido */
else ...
```

15

Abertura (Texto)

- ponteiro = **fopen**("filename","flags")
  - filename: nome do arquivo a ser aberto
  - flags: controla o modo de abertura
    - **r**: abre arquivo texto somente para leitura
    - **w**: cria arq. texto só para escrita (se já existe, é apagado)
    - **a**: anexa (append) ou cria arquivo texto só para escrita
    - **r+**: abre arquivo texto para leitura / escrita
    - **w+**: cria arq. texto para leitura / escrita (se já existe, é apagado)
    - **a+**: anexa ou cria arquivo texto para leitura / escrita

16

Abertura (Binário)

- ponteiro = **fopen**("filename","flags")
  - filename: nome do arquivo a ser aberto
  - flags: controla o modo de abertura
    - **rb**: abre arquivo binário somente para leitura
    - **wb**: cria arq. binário só para escrita (se já existe, é apagado)
    - **ab**: anexa (append) ou cria arquivo binário só para escrita
    - **r+b**: abre arquivo binário para leitura / escrita
    - **w+b**: cria arq. bin. para leitura / escrita (se já existe, é apagado)
    - **a+b**: anexa ou cria arquivo binário para leitura / escrita

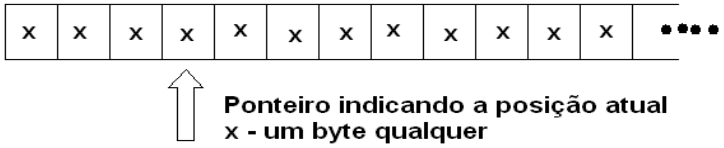
17

Fechamento

- **fclose**(ponteiro)
  - Descarrega conteúdo do *buffer*, garantindo que todas as informações sejam atualizadas e salvas
  - Encerra a associação entre arquivos lógico e físico
- Notas:
  - Em geral, S.O. fecha o arquivo se o programa não o fizer (prevenindo contra perdas de informação)
  - **fflush**(ponteiro) força descarregar o *buffer* sem precisar fechar o arquivo, prevenindo interrupções

18

O Ponteiro no Arquivo Lógico



Aponta para o primeiro byte a ser lido na próxima operação de leitura, ou substituído na próxima op. de escrita

19

Manipulação

- Principais Funções de L/E
  - Dados lidos/escritos um caractere por vez
    - caractere = **fgetc**(ponteiro)
    - **fputc**(caractere, ponteiro)
  - Exemplo: no quadro...

20

Manipulação

- Principais Funções de L/E (Strings)
  - **fgets**(str, no_max, ponteiro)
    - Lê caracteres do arquivo apontado por *ponteiro* e coloca na string *str*, até ler \n ou *no_max* caracteres
    - Se \n é lida, vira parte da string (diferente de **gets**)
    - \0 (NUL0 = ASCII 0) é incluído automaticamente ao final
  - **fputs**(str, ponteiro)
    - Escreve string *str* no arquivo apontado por *ponteiro*
  - Exemplo: no quadro...

21

Manipulação

- Principais Funções de L/E
  - Dados lidos/escritos de modo formatado
    - **fprintf**(ponteiro, "string_controle", variáveis)
    - **fscanf**(ponteiro, "string_controle", endereço(s))
      - Termina leitura de um número quando o primeiro caractere não numérico é encontrado
      - Termina leitura de string ao encontrar caractere 'branco' (espaço, tab ou \n). \0 é inserido ao final
  - Principais especificadores de formato
    - %d (inteiro), %f (float), %s (string), ...
  - Exemplo: no quadro...

22

Manipulação

- Principais Funções de L/E
  - Dados lidos/escritos como registros ou blocos de bytes
    - **fread**(endereço, tamanho, no elementos, ponteiro)
    - **fwrite**(endereço, tamanho, no elementos, ponteiro)
  - Exemplo:
    - **fwrite**(&v[0], sizeof(**int**), 10, pt_arq);

23

Manipulação

- Retornos das Principais Funções de L/E
  - **fgetc** / **fputc** : caractere lido/escrito ou **EOF**
  - **fgets**: ponteiro para a string lida ou ponteiro nulo (**NULL**)
  - **fputs**: inteiro qualquer diferente de zero ou **EOF**
  - **fread** / **fwrite**: número de *itens* lidos/escritos
  - **fscanf** / **fprintf**: no. de caracteres lidos/escritos ou **EOF**

24

Manipulação

- Controle de final de arquivo:
  - Via retornos de funções de leitura/escrita:
    - **EOF** (tipo especial): **fgetc**, **fscanf**, ...
    - Ponteiro **NULL**: **fgets**
    - No. de itens lidos/escritos: **fread**, **fwrite**
  - Via **feof**(ponteiro)
    - função com retorno lógico

25

Acesso Seqüencial vs Aleatório

- Leitura Seqüencial:
  - ponteiro de leitura avança byte a byte (ou por blocos), a partir de uma posição inicial
- Acesso Aleatório (Direto - Seeking):
  - acesso envolve o posicionamento do ponteiro em um byte ou registro arbitrário

26

Seeking

- Mover o ponteiro para certa posição no arquivo
- Em C:
  - bol = **fseek**(ponteiro, byte-offset, origem)
    - bol: retorno lógico (0 ou 1 - mal ou bem sucedido)
    - ponteiro: ponteiro arquivo
    - byte-offset: deslocamento, em bytes, a partir de *origem*
    - origem:
      - **SEEK_SET** (tipo especial – início do arquivo)
      - **SEEK_CUR** (tipo especial – posição atual)
      - **SEEK_END** (tipo especial – fim do arquivo)

27

Seeking

- Seeking (Nota):
  - As duas expressões abaixo são equivalentes
    - **fseek**(pt_arq, 0, SEEK_SET)
    - **rewind**(pt_arq)

28

Notas (Texto vs Binário)

- Todo arquivo, em última análise, é uma seqüência de bytes
- Ao contrário de Pascal, a linguagem C impõe pouca ou nenhuma restrição à manipulação de arquivos abertos como texto ou binário
- Funções como **fprintf** / **fscanf** e **fwrite** / **fread**, por exemplo, em princípio podem operar em ambos os casos
  - No caso de variáveis numéricas, a diferença é que as últimas as manipulam como em RAM, enquanto as primeiras como no console
- **Porém, tome cuidado...**

29

Notas (Texto vs Binário)

- Algumas funções podem operar de maneira diferente atuando sobre arquivos abertos como texto ou binário...
- Dependendo do sistema, a quebra de linha no arquivo pode ser interpretada como um par de bytes (CR + LF) em funções operando sobre um arq. aberto como **texto**
  - As funções **fgetc** e **fputc**, por exemplo, podem fazer conversões ao ler caracteres a partir de um arquivo aberto como **texto**
    - **fgetc**: Ao ler o caractere CR (ASCII 13), verifica se o próximo é LF (ASCII 10) e, se for, descarta o primeiro e retorna apenas LF
    - **fputc**: Ao escrever o caractere LF (ASCII 10), na verdade escreve dois bytes (CR + LF) no arquivo...

30

Exercício

- Analise ambos os códigos a seguir
- Execute-os em gcc para DOS/Windows e observe as diferenças na saída quando "texto.txt" é o arquivo texto em Notepad exemplificado anteriormente. Explique !

```
#include <stdio.h>

void main(void) {
    char ch;
    FILE *pt_arq;
    pt_arq = fopen("texto.txt","r");
    while ( (ch = getc(pt_arq)) != EOF )
            printf("Codigo ASCII: %d\n", ch);
    fclose(pt_arq);
    getchar();
}
```

```
#include <stdio.h>

void main(void) {
    char ch;
    FILE *pt_arq;
    pt_arq = fopen("texto.txt","rb");
    while ( (ch = getc(pt_arq)) != EOF )
            printf("Codigo ASCII: %d\n", ch);
    fclose(pt_arq);
    getchar();
}
```

Exercício

- Faça um programa C que use **fscanf** para ler um arquivo texto com duas colunas de números. Em cada linha do arquivo, os dois números devem ser separados por uma tabulação (tab). Os números devem impressos na tela na forma que apareceriam se o arquivo fosse aberto com um editor de texto ASCII.
- Refaça o exercício acima, mas agora usando **fgets** para imprimir o arquivo linha por linha

Exercício

- Fazer um programa C que leia, um número de vezes determinado pelo usuário, as seguintes informações:
  - nome, idade, curso, código
- O código deve ser crescente, por ordem de entrada:
  - 0, 1, 2, ...
- As informações devem ser estruturadas em um registro e gravadas em um arquivo binário
- Em seguida, o usuário deve entrar com um código e o programa deve escrever na tela os dados do registro correspondente:
  - O registro deve ser acessado diretamente através do código

Bibliografia

- Schildt, H. "C Completo e Total", 3a. Edição, Pearson, 1997.
- Damas, L. "Linguagem C", 10a. Edição, LTC, 2007