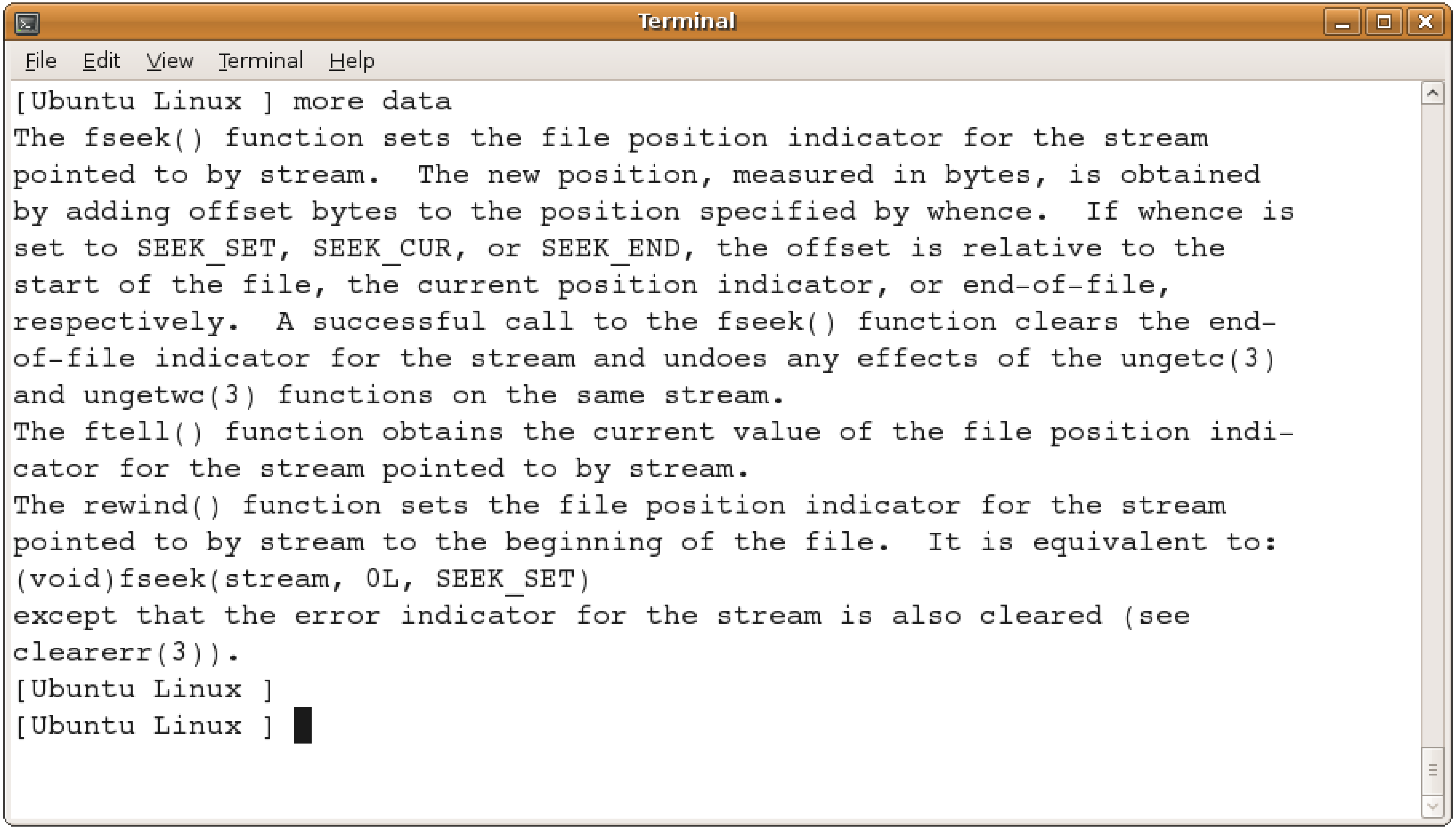
Sorting a File Line by Line

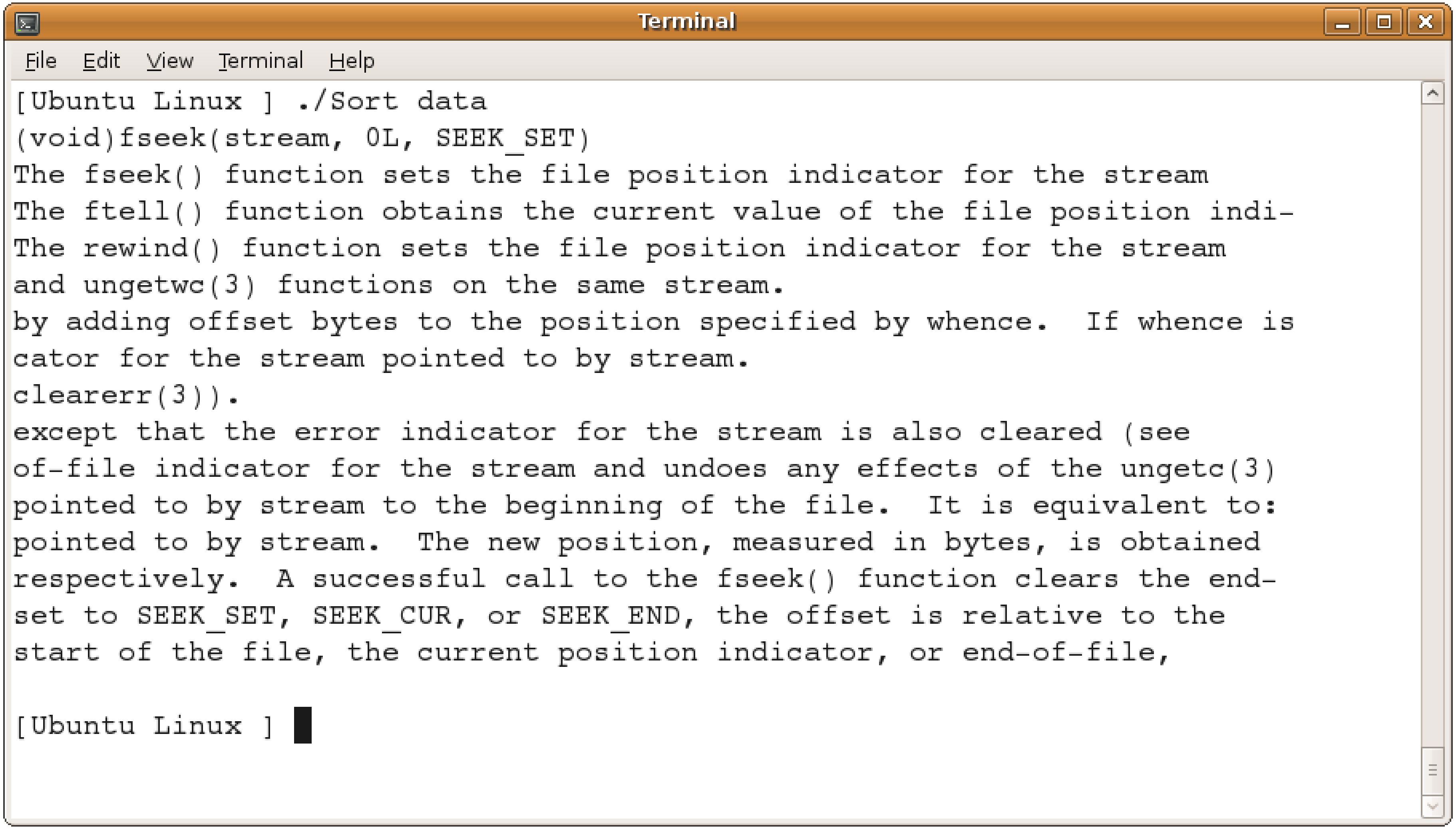
Yung-Hsiang Lu

Input



```
[Ubuntu Linux ] more data
The fseek() function sets the file position indicator for the stream
pointed to by stream.  The new position, measured in bytes, is obtained
by adding offset bytes to the position specified by whence.  If whence is
set to SEEK_SET, SEEK_CUR, or SEEK_END, the offset is relative to the
start of the file, the current position indicator, or end-of-file,
respectively.  A successful call to the fseek() function clears the end-
of-file indicator for the stream and undoes any effects of the ungetc(3)
and ungetwc(3) functions on the same stream.
The ftell() function obtains the current value of the file position indi-
cator for the stream pointed to by stream.
The rewind() function sets the file position indicator for the stream
pointed to by stream to the beginning of the file.  It is equivalent to:
(void)fseek(stream, 0L, SEEK_SET)
except that the error indicator for the stream is also cleared (see
clearerr(3)).
[Ubuntu Linux ]
[Ubuntu Linux ] 
```

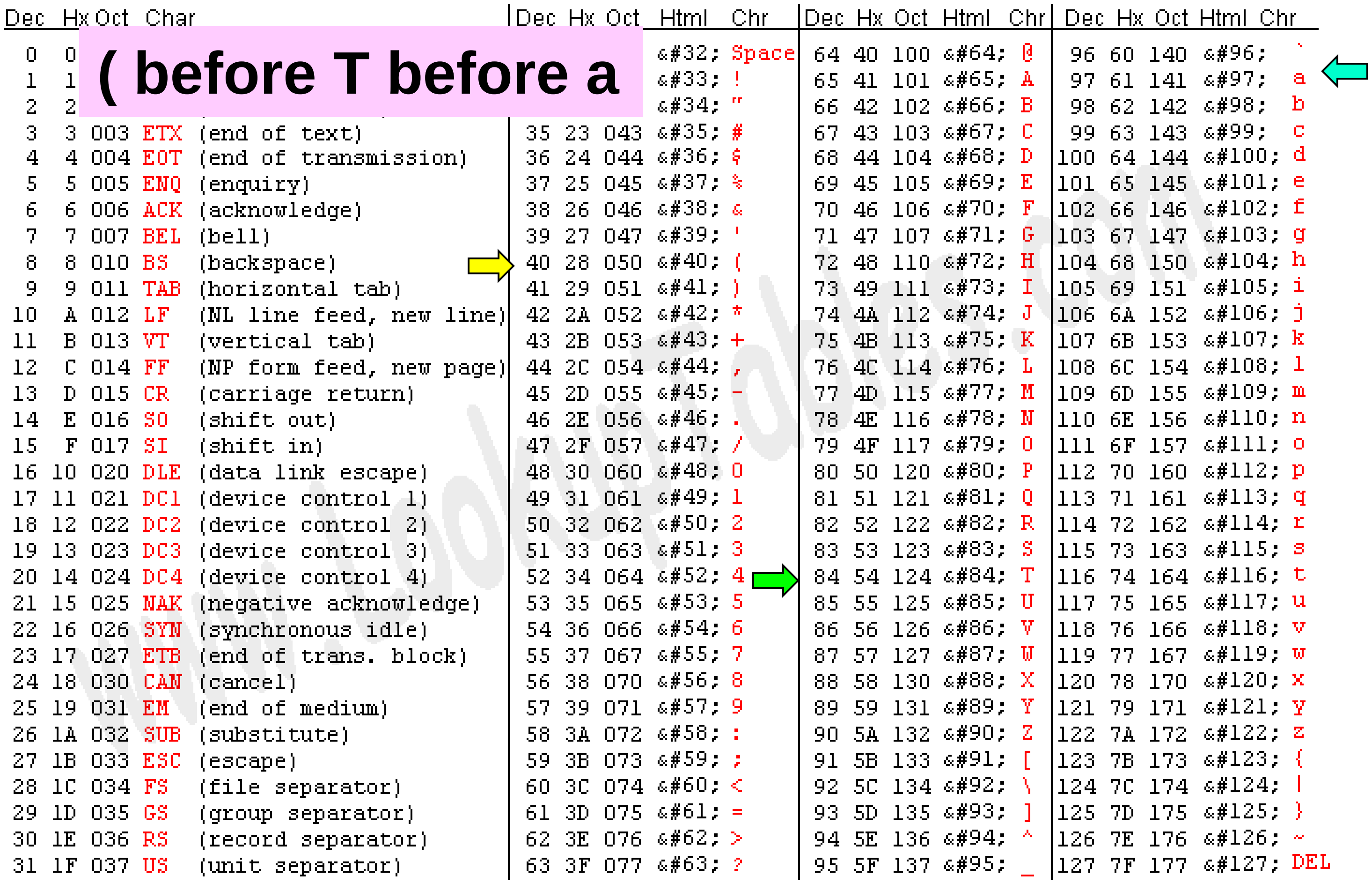
Output



```
[Ubuntu Linux ] ./Sort data
(void)fseek(stream, 0L, SEEK_SET)
The fseek() function sets the file position indicator for the stream
The ftell() function obtains the current value of the file position indi-
The rewind() function sets the file position indicator for the stream
and ungetwc(3) functions on the same stream.
by adding offset bytes to the position specified by whence.  If whence is
cator for the stream pointed to by stream.
clearerr(3)).
except that the error indicator for the stream is also cleared (see
of-file indicator for the stream and undoes any effects of the ungetc(3)
pointed to by stream to the beginning of the file.  It is equivalent to:
pointed to by stream.  The new position, measured in bytes, is obtained
respectively.  A successful call to the fseek() function clears the end-
set to SEEK_SET, SEEK_CUR, or SEEK_END, the offset is relative to the
start of the file, the current position indicator, or end-of-file,

[Ubuntu Linux ] 
```

ASCII Table



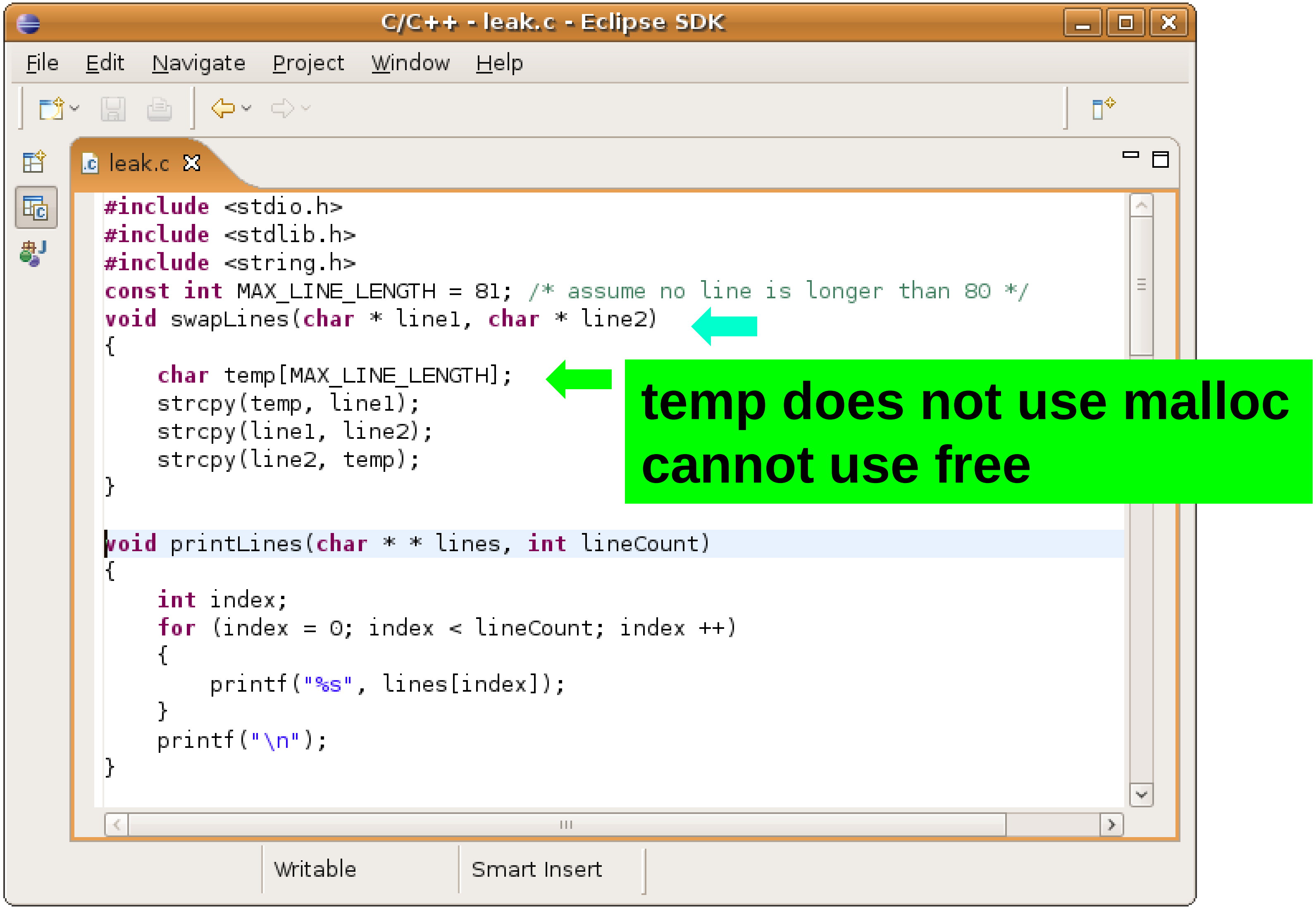
(before T before a

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

www.LookupTables.com

Source: www.LookupTables.com

swap strings



C/C++ - leak.c - Eclipse SDK
File Edit Navigate Project Window Help
leak.c
Writable
Smart Insert

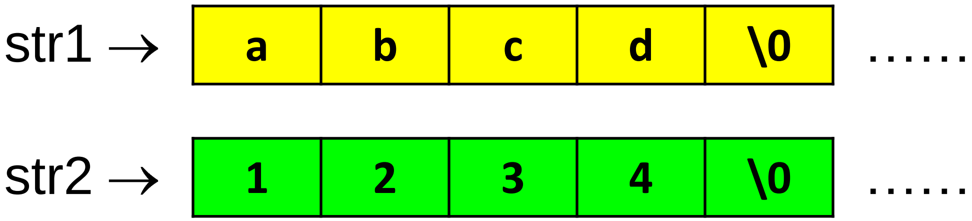
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
const int MAX_LINE_LENGTH = 81; /* assume no line is longer than 80 */
void swapLines(char * line1, char * line2)
{
    char temp[MAX_LINE_LENGTH];
    strcpy(temp, line1);
    strcpy(line1, line2);
    strcpy(line2, temp);
}

void printLines(char * * lines, int lineCount)
{
    int index;
    for (index = 0; index < lineCount; index ++)
    {
        printf("%s", lines[index]);
    }
    printf("\n");
}
```

temp does not use malloc
cannot use free

How to Swap Strings?

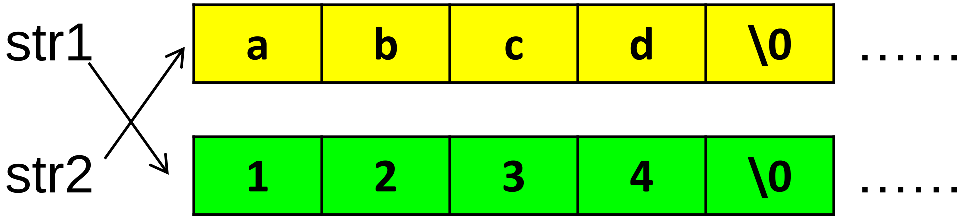
In C, a string is an array of char. An array is accessed through a pointer.



str1 → a b c d \0 ……

str2 → 1 2 3 4 \0 ……

swap by exchange the addresses ⇒



str1 a b c d \0 ……

str2 1 2 3 4 \0 ……

Swap Method 2

exchange the contents without changing the addresses

copy str1 to temp

The two strings must have enough memory for each other.

str1 →	a	b	c	d	\0
str2 →	1	2	3	4	\0
temp →	a	b	c	d	\0

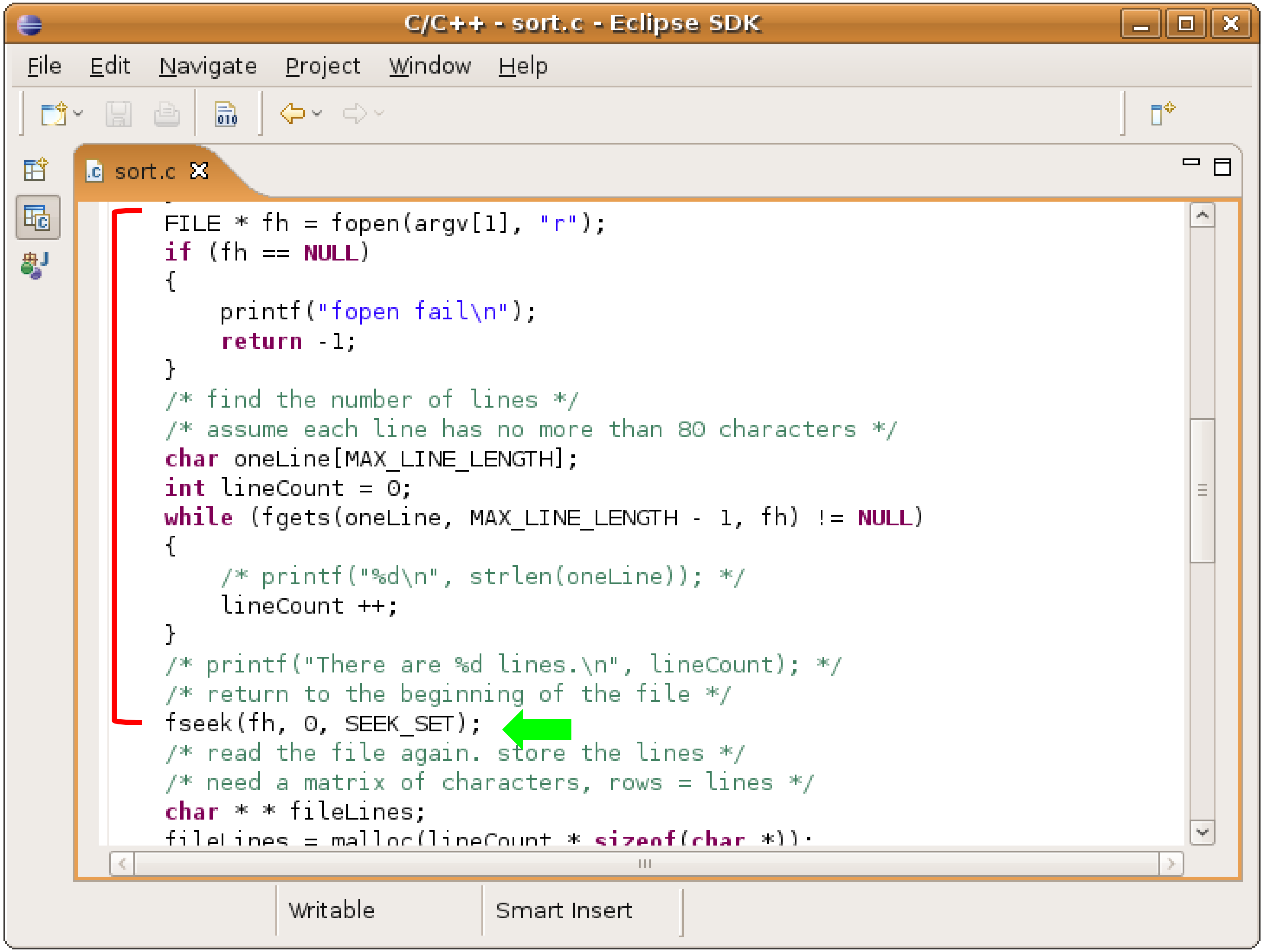
copy str2 to str1

str1 →	1	2	3	4	\0
str2 →	1	2	3	4	\0
temp →	a	b	c	d	\0

copy temp to str2

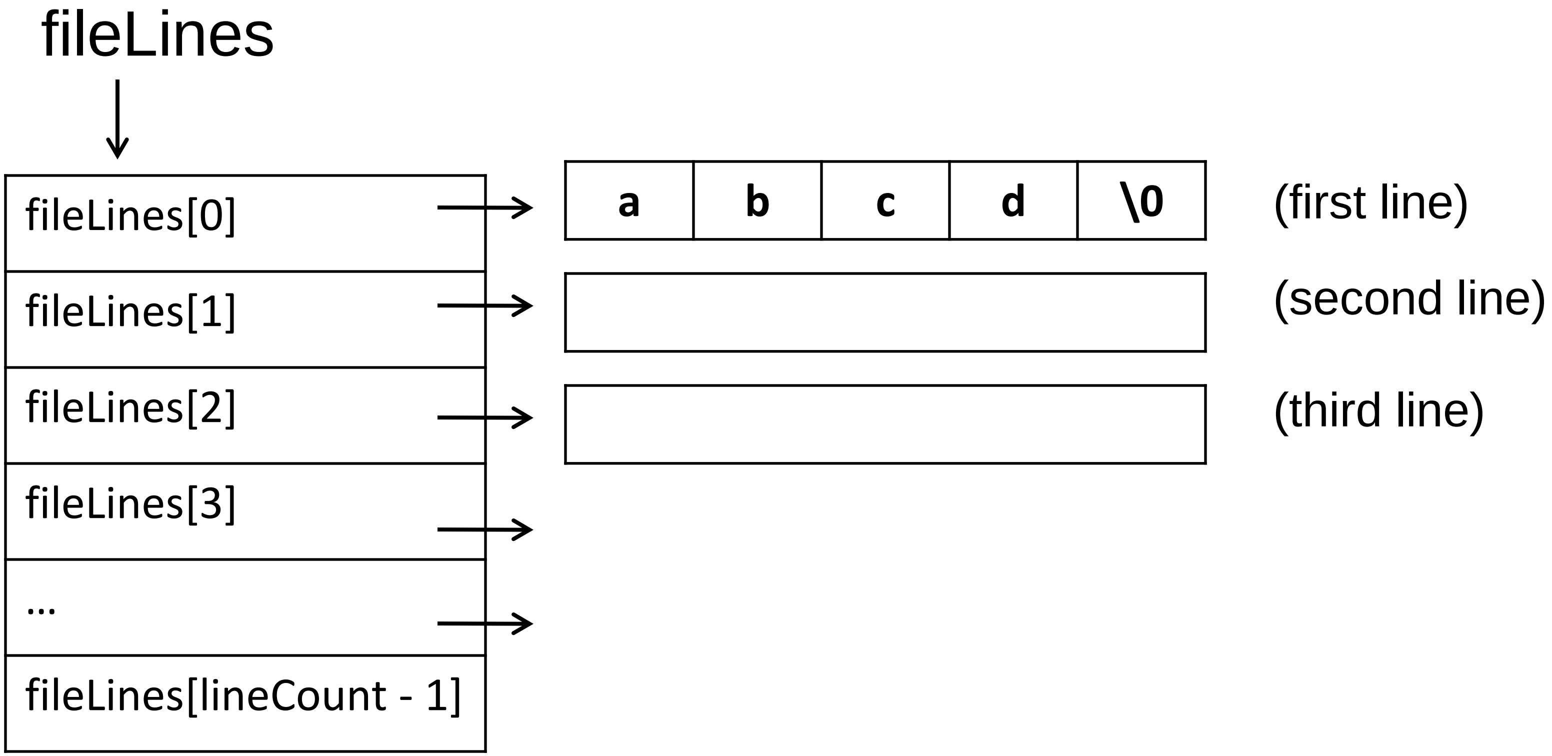
str1 →	1	2	3	4	\0
str2 →	a	b	c	d	\0
temp →	a	b	c	d	\0

Find the Number of Lines in a File



```
FILE * fh = fopen(argv[1], "r");
if (fh == NULL)
{
    printf("fopen fail\n");
    return -1;
}
/* find the number of lines */
/* assume each line has no more than 80 characters */
char oneLine[MAX_LINE_LENGTH];
int lineCount = 0;
while (fgets(oneLine, MAX_LINE_LENGTH - 1, fh) != NULL)
{
    /* printf("%d\n", strlen(oneLine)); */
    lineCount ++;
}
/* printf("There are %d lines.\n", lineCount); */
/* return to the beginning of the file */
fseek(fh, 0, SEEK_SET);
/* read the file again. store the lines */
/* need a matrix of characters, rows = lines */
char * * fileLines;
fileLines = malloc(lineCount * sizeof(char *));
```

Allocate Memory for the Lines



fileLines

fileLines[0]

fileLines[1]

fileLines[2]

fileLines[3]

…

fileLines[lineCount - 1]

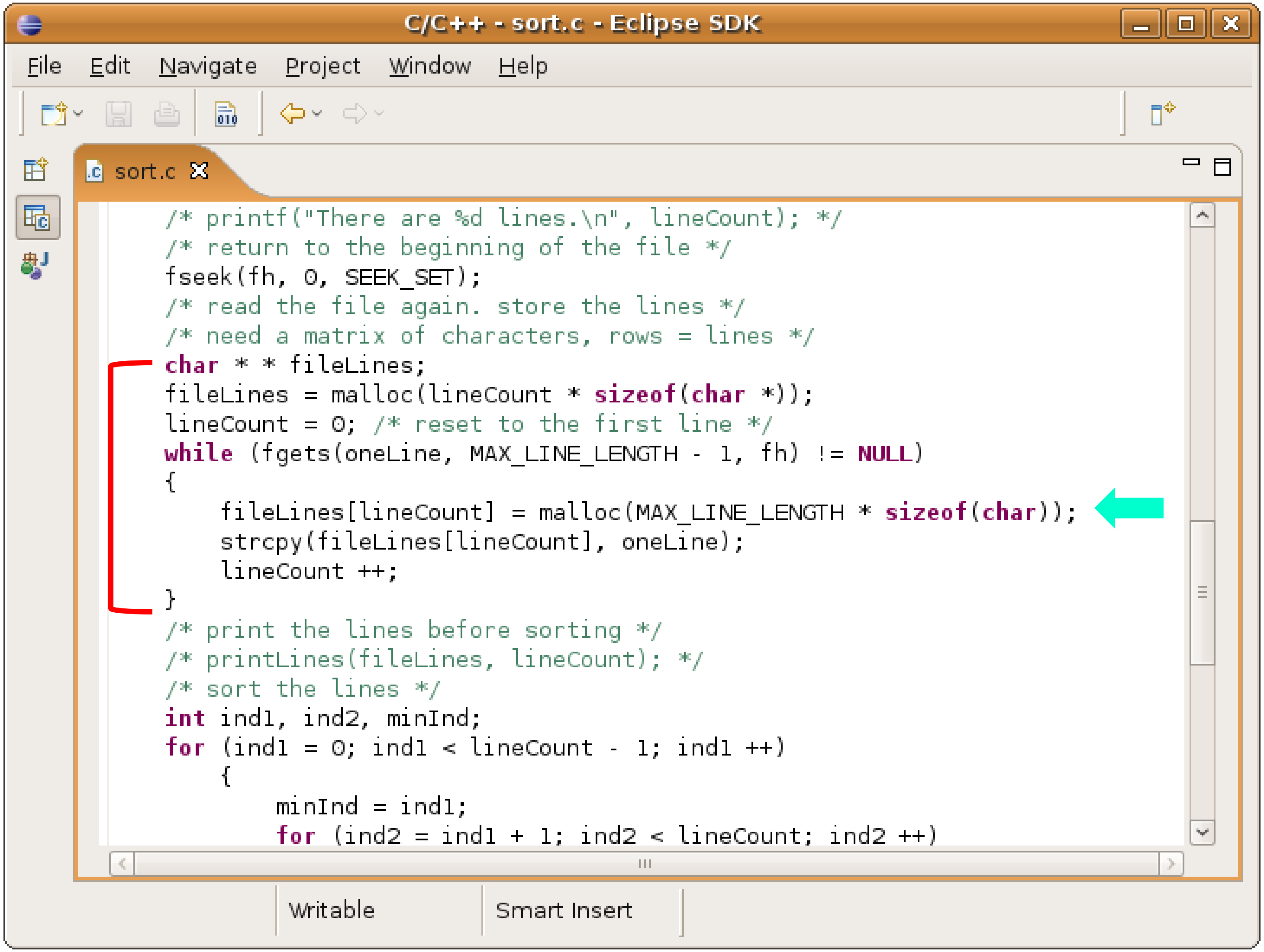
a	b	c	d	\0

(first line)

(second line)

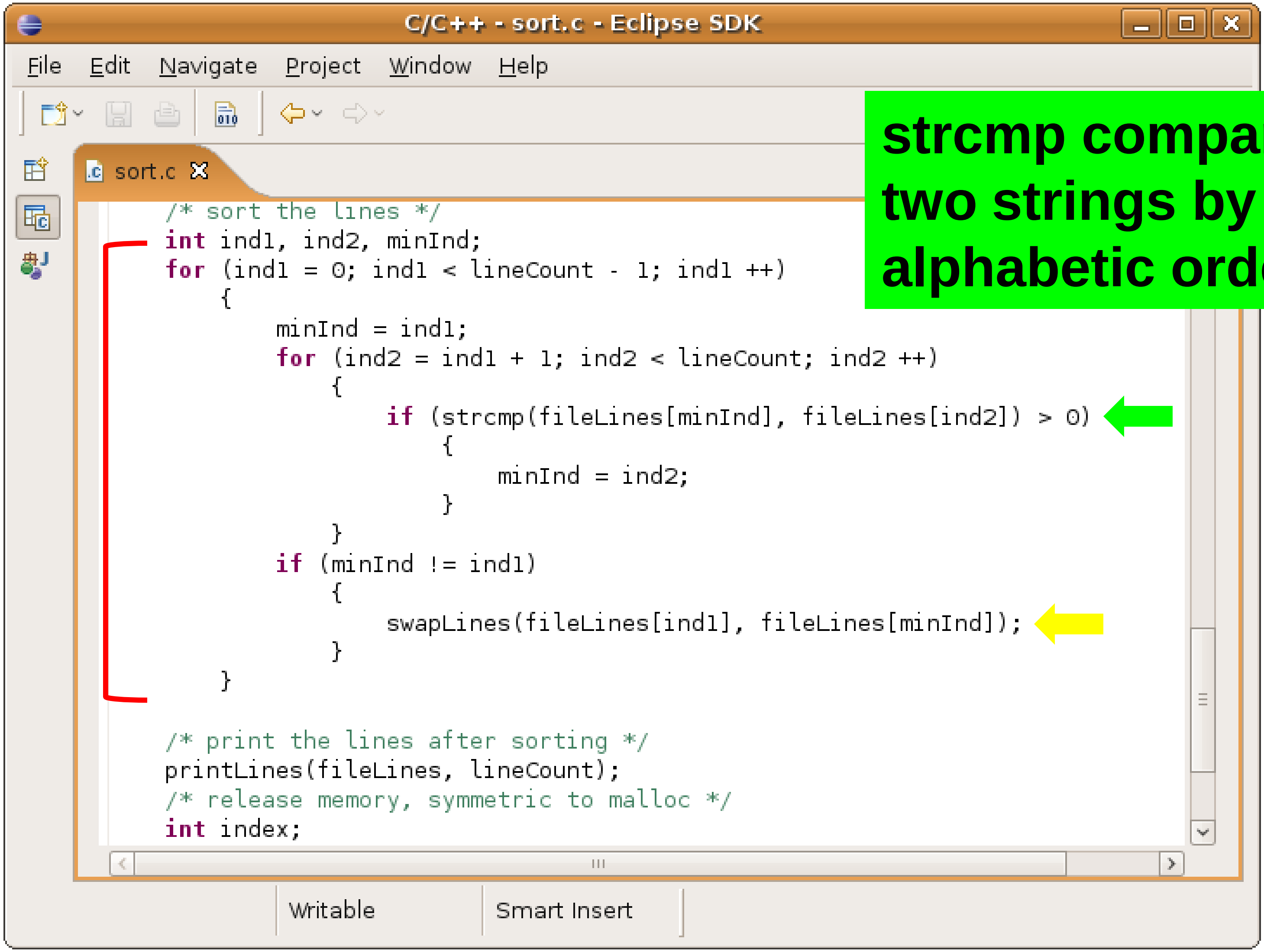
(third line)

Allocate Memory and Read Lines



```
/* printf("There are %d lines.\n", lineCount); */
/* return to the beginning of the file */
fseek(fh, 0, SEEK_SET);
/* read the file again. store the lines */
/* need a matrix of characters, rows = lines */
char * * fileLines;
fileLines = malloc(lineCount * sizeof(char *));
lineCount = 0; /* reset to the first line */
while (fgets(oneLine, MAX_LINE_LENGTH - 1, fh) != NULL)
{
    fileLines[lineCount] = malloc(MAX_LINE_LENGTH * sizeof(char));
    strcpy(fileLines[lineCount], oneLine);
    lineCount ++;
}
/* print the lines before sorting */
/* printLines(fileLines, lineCount); */
/* sort the lines */
int ind1, ind2, minInd;
for (ind1 = 0; ind1 < lineCount - 1; ind1 ++)
    {
        minInd = ind1;
        for (ind2 = ind1 + 1; ind2 < lineCount; ind2 ++)
```

Selection Sort



C/C++ - sort.c - Eclipse SDK
File Edit Navigate Project Window Help
sort.c
Writable
Smart Insert

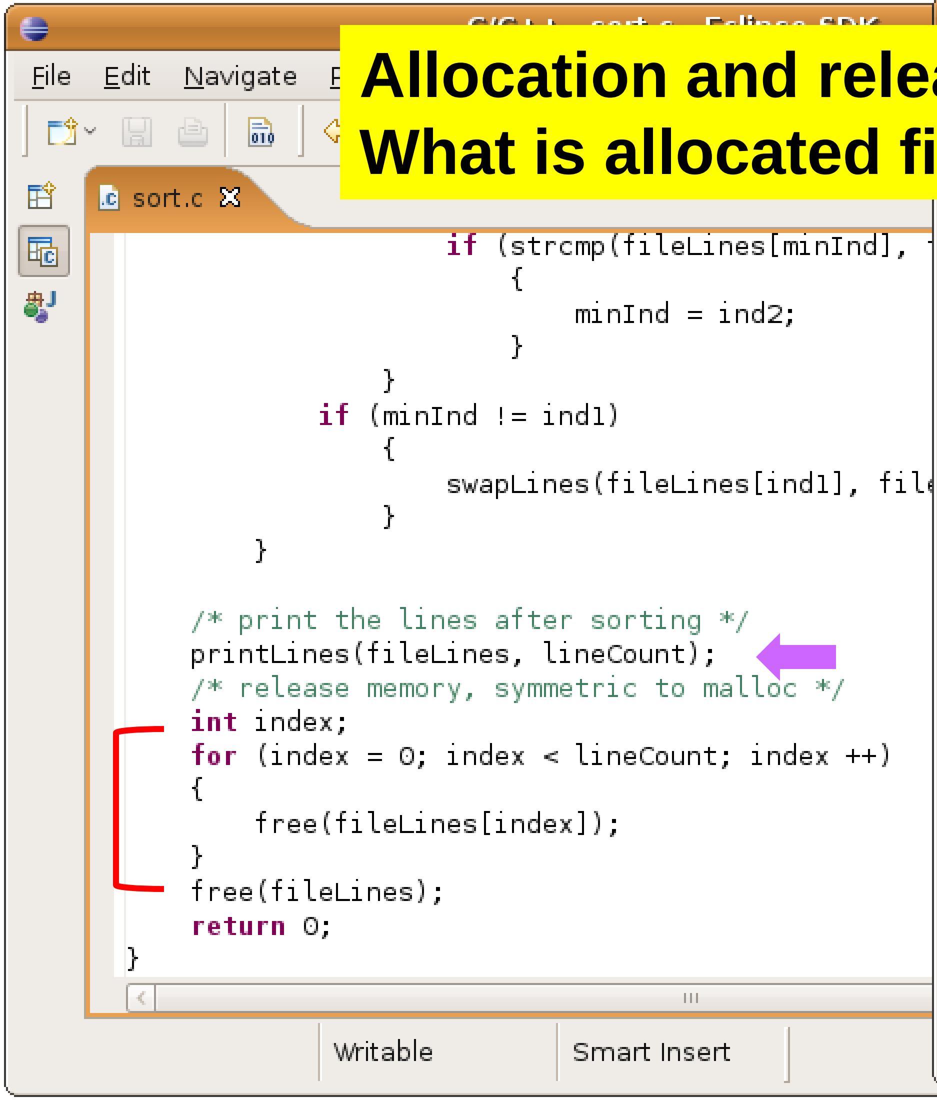
```
/* sort the lines */
int ind1, ind2, minInd;
for (ind1 = 0; ind1 < lineCount - 1; ind1 ++)
    {
        minInd = ind1;
        for (ind2 = ind1 + 1; ind2 < lineCount; ind2 ++)
            {
                if (strcmp(fileLines[minInd], fileLines[ind2]) > 0)
                    {
                        minInd = ind2;
                    }
            }
        if (minInd != ind1)
            {
                swapLines(fileLines[ind1], fileLines[minInd]);
            }
    }

/* print the lines after sorting */
printLines(fileLines, lineCount);
/* release memory, symmetric to malloc */
int index;
```

strcmp compares two strings by the alphabetic order

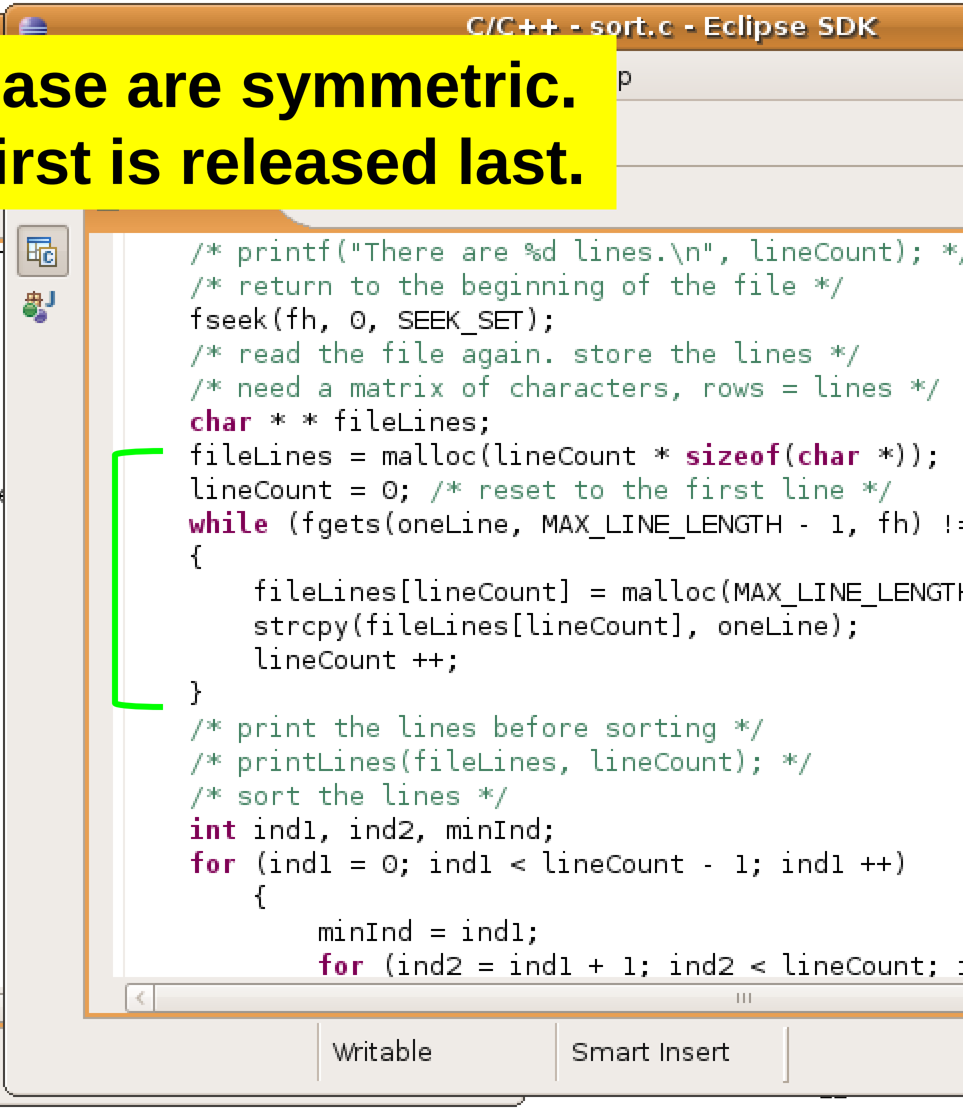
Release Memory

Allocation and release are symmetric.
What is allocated first is released last.



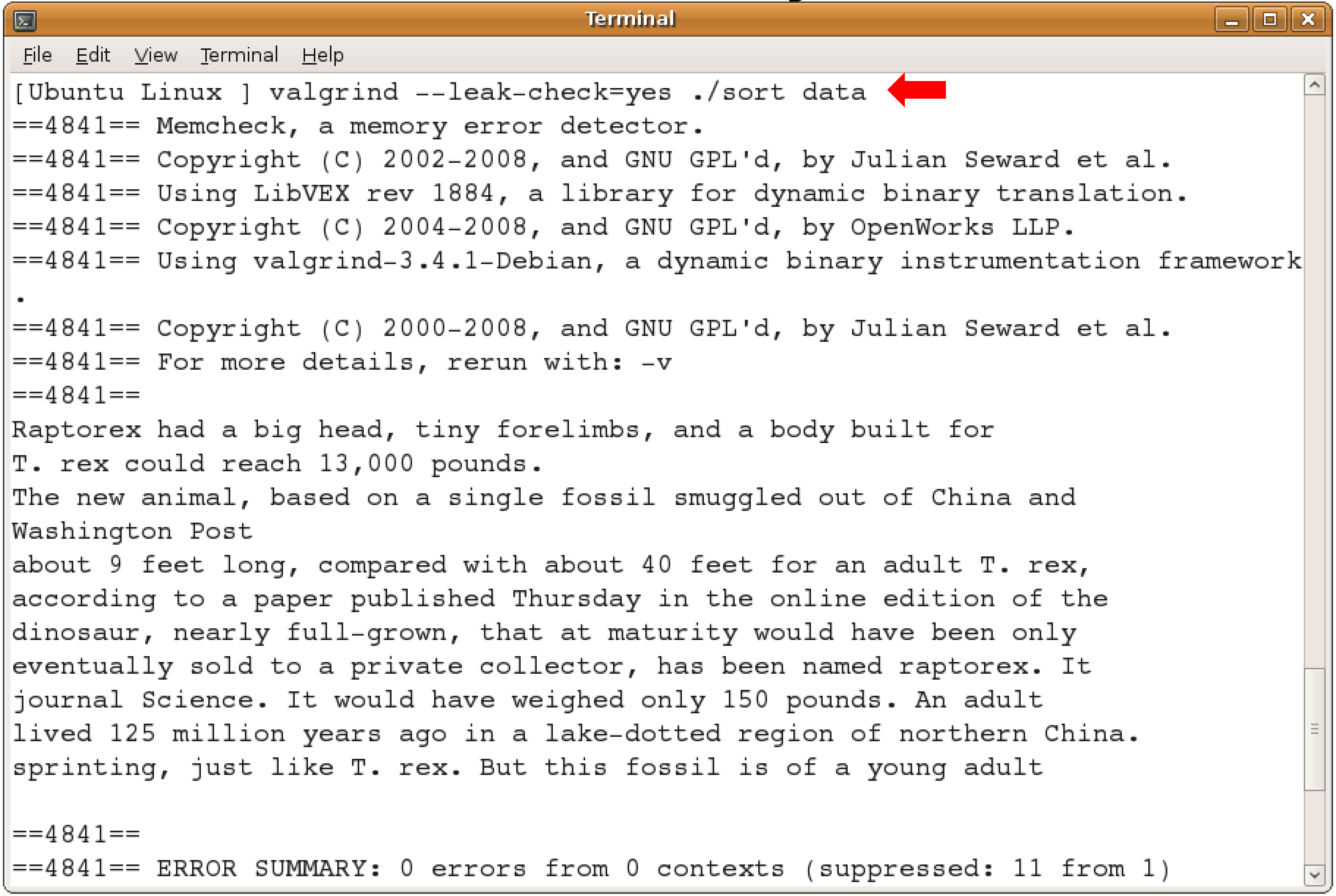
```
                    if (strcmp(fileLines[minInd], 
                        {
                            minInd = ind2;
                        }
                }
            if (minInd != ind1)
                {
                    swapLines(fileLines[ind1], file
                }
        }

    /* print the lines after sorting */
    printLines(fileLines, lineCount);
    /* release memory, symmetric to malloc */
    int index;
    for (index = 0; index < lineCount; index ++)
    {
        free(fileLines[index]);
    }
    free(fileLines);
    return 0;
}
```



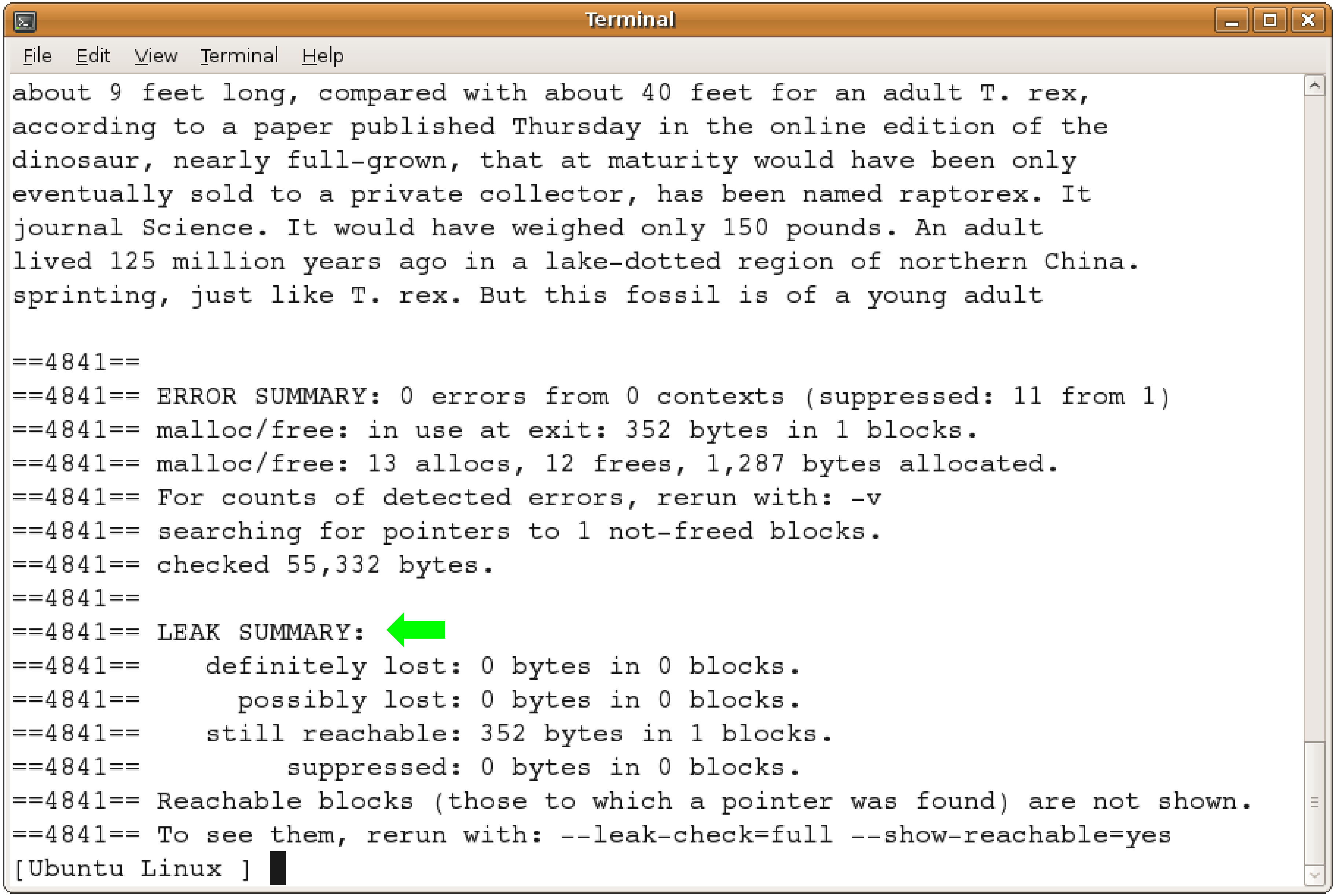
```
/* printf("There are %d lines.\n", lineCount); *
/* return to the beginning of the file */
fseek(fh, 0, SEEK_SET);
/* read the file again. store the lines */
/* need a matrix of characters, rows = lines */
char * * fileLines;
fileLines = malloc(lineCount * sizeof(char *));
lineCount = 0; /* reset to the first line */
while (fgets(oneLine, MAX_LINE_LENGTH - 1, fh) !=
{
    fileLines[lineCount] = malloc(MAX_LINE_LENGT
    strcpy(fileLines[lineCount], oneLine);
    lineCount ++;
}
/* print the lines before sorting */
/* printLines(fileLines, lineCount); */
/* sort the lines */
int ind1, ind2, minInd;
for (ind1 = 0; ind1 < lineCount - 1; ind1 ++)
    {
        minInd = ind1;
        for (ind2 = ind1 + 1; ind2 < lineCount; i
```

Check Memory Leak



```
[Ubuntu Linux ] valgrind --leak-check=yes ./sort data
==4841== Memcheck, a memory error detector.
==4841== Copyright (C) 2002-2008, and GNU GPL'd, by Julian Seward et al.
==4841== Using LibVEX rev 1884, a library for dynamic binary translation.
==4841== Copyright (C) 2004-2008, and GNU GPL'd, by OpenWorks LLP.
==4841== Using valgrind-3.4.1-Debian, a dynamic binary instrumentation framework
.
==4841== Copyright (C) 2000-2008, and GNU GPL'd, by Julian Seward et al.
==4841== For more details, rerun with: -v
==4841==
Raptorex had a big head, tiny forelimbs, and a body built for
T. rex could reach 13,000 pounds.
The new animal, based on a single fossil smuggled out of China and
Washington Post
about 9 feet long, compared with about 40 feet for an adult T. rex,
according to a paper published Thursday in the online edition of the
dinosaur, nearly full-grown, that at maturity would have been only
eventually sold to a private collector, has been named raptorex. It
journal Science. It would have weighed only 150 pounds. An adult
lived 125 million years ago in a lake-dotted region of northern China.
sprinting, just like T. rex. But this fossil is of a young adult

==4841==
==4841== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 11 from 1)
```



```
about 9 feet long, compared with about 40 feet for an adult T. rex,
according to a paper published Thursday in the online edition of the
dinosaur, nearly full-grown, that at maturity would have been only
eventually sold to a private collector, has been named raptorex. It
journal Science. It would have weighed only 150 pounds. An adult
lived 125 million years ago in a lake-dotted region of northern China.
sprinting, just like T. rex. But this fossil is of a young adult

==4841==
==4841== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 11 from 1)
==4841== malloc/free: in use at exit: 352 bytes in 1 blocks.
==4841== malloc/free: 13 allocs, 12 frees, 1,287 bytes allocated.
==4841== For counts of detected errors, rerun with: -v
==4841== searching for pointers to 1 not-freed blocks.
==4841== checked 55,332 bytes.
==4841==
==4841== LEAK SUMMARY:
==4841==    definitely lost: 0 bytes in 0 blocks.
==4841==      possibly lost: 0 bytes in 0 blocks.
==4841==    still reachable: 352 bytes in 1 blocks.
==4841==         suppressed: 0 bytes in 0 blocks.
==4841== Reachable blocks (those to which a pointer was found) are not shown.
==4841== To see them, rerun with: --leak-check=full --show-reachable=yes
[Ubuntu Linux ] 
```