"Stupidity is while (1) { tryAgain(); }"

- Unknown

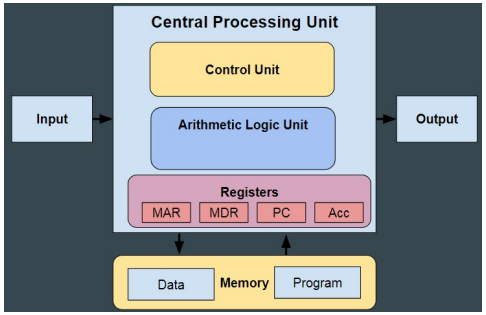
CSE102
Computer Programming with C

2020-2021 Spring Semester

Files

1

Von Neumann Architecture



2

File Processing

- Files: used for permanent storage of information
- Two types of files:
  - Text files
  - Binary files

3

Text Files

- Text file: collection of characters
  - Can be considered as stream of characters
    - Input stream (e.g., keyboard : stdin)
    - Output stream (e.g., screen : stdout, stderr)
  - Can be created by using editors
    - Readable by human
  - Special characters
    - New line character
    - End of file character (EOF is returned when read)
    - Other escape sequences

4

Escape Sequences

TABLE 12.1 Meanings of Common Escape Sequences

Escape Sequence	Meaning
'\n'	new line
'\t'	tab
'\f'	form feed (new page)
'\r'	return (go back to column 1 of current output line)
'\b'	backspace

5

Formatting Output with `printf`

TABLE 12.2 Placeholders for printf Format Strings

Placeholder	Used for Output of	Example	Output
%c	a single character	printf("%c%c%c\n", 'a', '\n', 'b');	a b
%s	a string	printf("%s%s%s\n", "Hi, how ", "are you?");	Hi, how are you?
%d	an integer (in base 10)	printf("%d\n", 43);	43
%o	an integer (in base 8)	printf("%o\n", 43);	53
%x	an integer (in base 16)	printf("%x\n", 43);	2b
%f	a floating-point number	printf("%f\n", 81.97);	81.970000
%e	a floating-point number in scientific notation	printf("%e\n", 81.97);	8.197000e+01
%E	a floating point number in scientific notation	printf("%E\n", 81.97);	8.197000E+01
%%	a single % sign	printf("%d%%\n", 10);	10%

6

Formatting Output with `printf`

TABLE 12.3 Designating Field Width, Justification, and Precision in Format Strings

Example	Meaning of Highlighted Format String Fragment	Output Produced
printf("%5d%4d\n", 100, 2);	Display an integer right-justified in a field of 5 columns.	␣␣100␣␣␣2
printf ("%2d with label\n", 5210);	Display an integer in a field of 2 columns. *Note:* Field is too small.	5210␣with␣label
printf("%-16s%d\n", "Jeri R. Hanly", 28);	Display a string left-justified in a field of 16 columns.	Jeri␣R.␣Hanly␣␣␣28
printf("%15f\n", 981.48);	Display a floating-point number right-justified in a field of 15 columns.	␣␣␣␣␣981.480000
printf("%10.3f\n", 981.48);	Display a floating-point number right-justified in a field of 10 columns, with 3 digits to the right of the decimal point.	␣␣␣␣981.480
printf("%7.1f\n", 981.48);	Display a floating-point number right-justified in a field of 7 columns, with 1 digit to the right of the decimal point	␣␣␣981.5
printf("%12.3e\n", 981.40);	Display a floating-point number in scientific notation right-justified in a field of 12 columns, with 3 digits to the right of the decimal point and a lowercase e before the exponent.	␣␣␣9.815e+02
printf("%.5R\n", 0.090140);	Display a floating-point number in scientific notation, with 5 digits to the right of the decimal point and an uppercase E before the exponent.	9.81480E-02

7

File Pointer

- Allows to access a file

```
FILE *fileptr;
fileptr = fopen("filename", "access mode");
if (fileptr == NULL)
   printf("File open error");
else
   …. process file ….
fclose(fileptr);
```

- Processing with `getc`, `putc`, `fscanf` and `fprintf`
  - What if `stdin` or `stdout` is used as `FILE *`

8

Copying a Text File

```
/*
 *  Makes a backup file.  Repeatedly prompts for the name of a file to
 *  back up until a name is provided that corresponds to an available
 *  file.  Then it prompts for the name of the backup file and creates
 *  the file copy.
 */

#include <stdio.h>
#define  STRSIZ 80

int
main(void)
{
      char  in_name[STRSIZ],    /* strings giving names                    */
            out_name[STRSIZ];   /*    of input and backup files           */
      FILE *inp,                /* file pointers for input and             */
           *outp;               /*    backup files                        */
      char ch;                  /* one character of input file             */

      /*  Get the name of the file to back up and open the file for input   */
      printf("Enter name of file you want to back up> ");
```

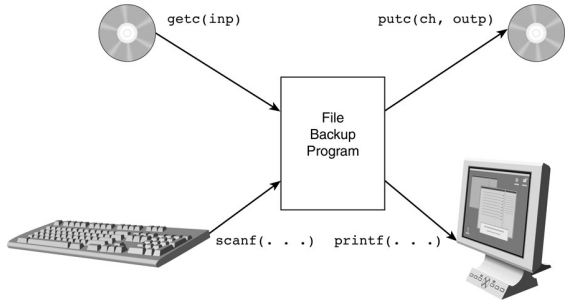
(continued)

9

```
      for (scanf("%s", in_name);
           (inp = fopen(in_name, "r")) == NULL;
           scanf("%s", in_name)) {
         printf("Cannot open %s for input\n", in_name);
         printf("Re-enter file name> ");
      }

      /*  Get name to use for backup file and open file for output          */
      printf("Enter name for backup copy> ");
      for (scanf("%s", out_name);
           (outp = fopen(out_name, "w")) == NULL;
           scanf("%s", out_name)) {
         printf("Cannot open %s for output\n", out_name);
         printf("Re-enter file name> ");
      }

      /*  Make backup copy one character at a time                          */
      for  (ch = getc(inp);  ch != EOF;  ch = getc(inp))
          putc(ch, outp);

      /*  Close files and notify user of backup completion                  */
      fclose(inp);
      fclose(outp);
      printf("Copied %s to %s.\n", in_name, out_name);

      return(0);
}
```


10

Input and Output Streams



11

Binary Files

- Binary Files stores the data in their internal representation
  - Note that Text files stores the data as character sequence
    - requires conversion between data types and stream of characters
  - No conversion in binary files
    - Higher performance
    - Less storage
    - Higher precision for doubles
  - System dependent
    - Not portable
  - Not human readable

12

Binary Files

```
FILE *fileptr;
fileptr = fopen("filename", "access mode");
if (fileptr == NULL)
  printf("File open error");
else
  …. process file ….
fclose(fileptr);
```

- Access more is "rb" or "wb"
- Processing with fwrite or fread
  - Ex: creating a binary file of integer

Creating a Binary File of Integers

```
1. FILE *binaryp;
2. int   i;
3.
4. binaryp = fopen("nums.bin", "wb");
5.
6. for  (i = 2;   i <= 500;   i += 2)
7.     fwrite(&i, sizeof (int), 1, binaryp);
8.
9. fclose(binaryp);
```


fread and fwrite

```
fwrite(pointer, size_of_component, num_of_values, fileptr)

day_t a[20];
fwrite (a, sizeof(day_t), 20, bptr);

int fread(pointer, size_of_component, num_of_values, fileptr)

int a[20];
num = fread(a, sizeof(int), 20, bptr);
```


Text file vs Binary file

- Assume following declarations

```
#define STRSIZ 10
#define MAX 40

typedef struct {
        char    name[20];
        double          diameter;
        int     moons;
        double          orbit_time,
                rotation_time;
} planet_t;

double nums[MAX], data;
planet_t a_planet;
int i, n, status;
FILE *plan_bin_inp, *plan_bin_outp, *plan_txt_inp, *plan_txt_outp;
FILE *doub_bin_inp, *doub_bin_outp, *doub_txt_inp, *doub_txt_outp;
```

TABLE 12.5 *Data I/O Using Text and Binary Files*

Example	Text File I/O	Binary File I/O	Purpose
1	`plan_txt_inp = fopen("planets.txt", "r");` `doub_txt_inp = fopen("nums.txt", "r");`	`plan_bin_inp = fopen("planets.bin", "rb");` `doub_bin_inp = fopen("nums.bin", "rb");`	Open for input a file of planets and a file of numbers, saving file pointers for use in calls to input functions.
2	`plan_txt_outp = fopen("pl_out.txt", "w");` `doub_txt_outp = fopen("nm_out.txt", "w");`	`plan_bin_outp = fopen("pl_out.bin", "wb");` `doub_bin_outp = fopen("nm_out.bin", "wb");`	Open for output a file of planets and a file of numbers, saving file pointers for use in calls to output functions.
3	`fscanf(plan_txt_inp, "%s%lf%d%lf%lf", a_planet.name, &a_planet.diameter, &a_planet.moons, &a_planet.orbit_time, &a_planet.rotation_time);`	`fread(&a_planet, sizeof (planet_t), 1, plan_bin_inp);`	Copy one planet structure into memory from the data file.
4	`fprintf(plan_txt_outp, "%s %e %d %e %e", a_planet.name, a_planet.diameter, a_planet.moons, a_planet.orbit_time, a_planet.rotation_time);`	`fwrite(&a_planet, sizeof (planet_t), 1, plan_bin_outp);`	Write one planet structure to the output file.

17

TABLE 12.5 (continued)

Example	Text File I/O	Binary File I/O	Purpose
5	`for (i = 0; i < MAX; ++i) fscanf(doub_txt_inp, "%lf", &nums[i]);`	`fread(nums, sizeof (double), MAX, doub_bin_inp);`	Fill array `nums` with type `double` values from input file.
6	`for (i = 0; i < MAX; ++i) fprintf(doub_txt_outp, "%e\n", nums[i]);`	`fwrite(nums, sizeof (double), MAX, doub_bin_outp);`	Write contents of array `nums` to output file.
7	`n = 0; for (status = fscanf(doub_txt_inp, "%lf", &data); status != EOF && n < MAX; status = fscanf(doub_txt_inp, "%lf", &data)) nums[n++] = data;`	`n = fread(nums, sizeof (double), MAX, doub_bin_inp);`	Fill `nums` with data until `EOF` encountered, setting `n` to the number of values stored.
8	`fclose(plan_txt_inp); fclose(plan_txt_outp); fclose(doub_txt_inp); fclose(doub_txt_outp);`	`fclose(plan_bin_inp); fclose(plan_bin_outp); fclose(doub_bin_inp); fclose(doub_bin_outp);`	Close all input and output files.

18

Case Study: Database Inquiry Problem

- Database
  - File
    - Record
      - Field
- Inventory database
  - Inventory file
    - Product record
      - Stock number
      - Category
      - Technical description
      - Price

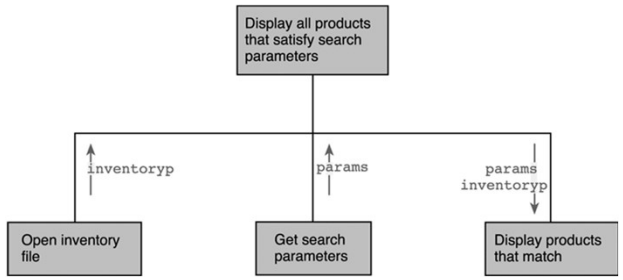
19

Case Study: Database Inquiry Problem

- Possible queries:
  - What printer stands that cost less than $100 are available?
  - What product has the code 5432?
  - What types of data cartridges are available?
- Analysis:
  - Open inventory file
  - Get search parameters
  - Display products that satisfy the search parameters

20

Case Study: Database Inquiry Problem



21

```
/*
 *  Displays all products in the database that satisfy the search
 *  parameters specified by the program user.
 */
#include <stdio.h>
#include <string.h>

#define MIN_STOCK    1111    /* minimum stock number                          */
#define MAX_STOCK    9999    /* maximum stock number                          */
#define MAX_PRICE    1000.00 /* maximum product price                         */
#define STR_SIZ      80      /* number of characters in a string              */

typedef struct {             /* product structure type                        */
      int    stock_num;          /* stock number                              */
      char   category[STR_SIZ];
      char   tech_descript[STR_SIZ];
      double price;
} product_t;

typedef struct {             /* search parameter bounds type                  */
      int    low_stock, high_stock;
      char   low_category[STR_SIZ], high_category[STR_SIZ];
      char   low_tech_descript[STR_SIZ], high_tech_descript[STR_SIZ];
      double low_price, high_price;
} search_params_t;

search_params_t get_params(void);
void display_match(FILE *databasep, search_params_t params);

/* Insert prototypes of functions needed by get_params and display_match */
```


22

```
int
main(void)
{
      char            inv_filename[STR_SIZ];  /* name of inventory file       */
      FILE           *inventoryp;             /* inventory file pointer       */
      search_params_t params;                 /* search parameter bounds      */

      /*  Get name of inventory file and open it                              */
      printf("Enter name of inventory file> ");
      scanf("%s", inv_filename);
      inventoryp = fopen(inv_filename, "rb");

      /*  Get the search parameters                                           */
      params = get_params();

      /*  Display all products that satisfy the search parameters             */
      display_match(inventoryp, params);

      return(0);
}
```


23

```
/*
 *  Prompts the user to enter the search parameters
 */
search_params_t
get_params(void)
{
   /* body of get_params to be inserted */
}
 /*
  *  Displays records of all products in the inventory that satisfy search
  *  parameters.
  *  Pre:  databasep accesses a binary file of product_t records that has
  *        been opened as an input file, and params is defined
  */
 void
 display_match(FILE           *databasep, /* input - file pointer to binary
                                                 database file
 */
               search_params_t params)    /* input - search parameter bounds
                                                 */
 {
   /* body of display_match to be inserted */
 }

 /*  Insert functions needed by get_params and display_match
 */
```

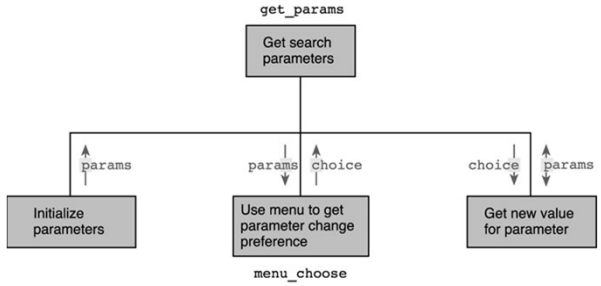

24

Algorithm for `get_params`

1. Initialize params to permit widest possible search
2. Display menu and get response to store in choice
3. Repeat while the choice is not 'q'
   4. Select appropriate prompt and get parameter value
   5. Display menu and get response to store in choice
6. Return search parameters

25

Structure Chart for `get_params`



26

```
Select by letter a search parameter to set, or enter q to
accept parameters shown.
      Search Parameter                          Current Value
[a]  Low bound for stock number                 1111
[b]  High bound for stock number                9999
[c]  Low bound for category                     aaaa
[d]  High bound for category                    zzzz
[e]  Low bound for technical description        aaaa
[f]  High bound for technical description       zzzz
[g]  Low bound for price                        $    0.00
[h]  High bound for price                       $1000.00

Selection> c
New low bound for category> modem

Select by letter a search parameter to set, or enter q to accept
parameters shown.
      Search Parameter                          Current Value
[a]  Low bound for stock number                 1111
[b]  High bound for stock number                9999
[c]  Low bound for category                     modem
[d]  High bound for category                    zzzz
[e]  Low bound for technical description        aaaa
[f]  High bound for technical description       zzzz
[g]  Low bound for price                        $    0.00
[h]  High bound for price                       $1000.00
```


27

```
Selection> d
New high bound for category> modem

Select by letter a search parameter to set, or enter q to accept
parameters shown.
      Search Parameter                          Current Value
[a]  Low bound for stock number                 1111
[b]  High bound for stock number                9999
[c]  Low bound for category                     modem
[d]  High bound for category                    modem
[e]  Low bound for technical description        aaaa
[f]  High bound for technical description       zzzz
[g]  Low bound for price                        $    0.00
[h]  High bound for price                       $1000.00

Selection> h
New high bound for price> 199.99
```


28

```
Select by letter a search parameter to set, or enter q to accept
parameters shown.
      Search Parameter                                  Current Value
[a]  Low bound for stock number                         1111
[b]  High bound for stock number                        9999
[c]  Low bound for category                             modem
[d]  High bound for category                            modem
[e]  Low bound for technical description                aaaa
[f]  High bound for technical description               zzzz
[g]  Low bound for price                                $    0.00
[h]  High bound for price                               $ 199.99

Selection> q
```


29

```
1.  /*
2.   *  Displays a lettered menu with the current values of search parameters.
3.   *  Returns the letter the user enters.  A letter in the range a..h selects
4.   *  a parameter to change; q quits, accepting search parameters shown.
5.   *  Post:  first non whitespace character entered is returned
6.   */
7.  char
8.  menu_choose(search_params_t params) /* input - current search parameter
9.                                                  bounds                    */
10. {
11.       char choice;
12.
13.       printf("Select by letter a search parameter to set or enter ");
14.       printf("q to\naccept parameters shown.\n\n");
15.       printf("      Search parameter                    ");
16.       printf("Current value\n\n");
17.       printf("[a]  Low bound for stock number            %4d\n",
18.              params.low_stock);
19.       printf("[b]  High bound for stock number           %4d\n",
20.              params.high_stock);
21.       printf("[c]  Low bound for category                %s\n",
22.              params.low_category);
```

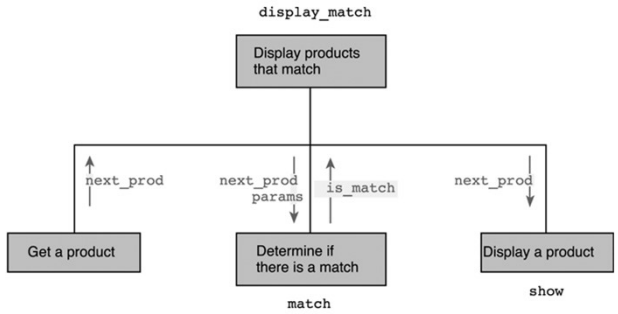
(continued)

30

```
23.       printf("[d]  High bound for category               %s\n",
24.              params.high_category);
25.       printf("[e]  Low bound for technical description   %s\n",
26.              params.low_tech_descript);
27.       printf("[f]  High bound for technical description  %s\n",
28.              params.high_tech_descript);
29.       printf("[g]  Low bound for price                   $%7.2f\n",
30.              params.low_price);
31.       printf("[h]  High bound for price                  $%7.2f\n\n",
32.              params.high_price);
33.
34.       printf("Selection> ");
35.       scanf(" %c", &choice);
36.
37.       return (choice);
38. }
39.
```


31

Structure Chart for display_match



32

```
39.
40. /*
41.  *  Determines whether record prod satisfies all search parameters
42.  */
43. int
44. match(product_t       prod,    /*  input - record to check         */
45.       search_params_t params)  /*  input - parameters to satisfy   */
46. {
47.       return (strcmp(params.low_category, prod.category) <= 0            &&
48.               strcmp(prod.category, params.high_category) <= 0           &&
49.               strcmp(params.low_tech_descript, prod.tech_descript) <= 0  &&
50.               strcmp(prod.tech_descript, params.high_tech_descript) <= 0 &&
51.               params.low_price <= prod.price                             &&
52.               prod.price <= params.high_price);
53. }
54. /*
55.  *  *** STUB ***
56.  *  Displays each field of prod.  Leaves a blank line after the product
57.  *  display.
58.  */
59. void
60. show(product_t prod)
61. {
62.       printf("Function show entered with product number %d\n",
63.              prod.stock_num);
64. }
65.
66.
```


33

```
67. /*
68.  *  Displays records of all products in the inventory that satisfy search
69.  *  parameters.
70.  *  Pre:  databasep accesses a binary file of product_t records that has
71.  *        been opened as an input file, and params is defined
72.  */
73. void
74. display_match(FILE           *databasep,  /*  file pointer to binary
75.                                             database file                 */
76.                search_params_t params)     /*  input - search parameter bounds  */
77. {
78.       product_t next_prod;       /*  current product from database          */
79.       int       no_matches = 1;  /*  flag indicating if no matches have
80.                                      been found                            */
81.       int       status;          /*  input file status                      */
82.
83.       /*  Advances to first record with a stock number greater than or
84.           equal to lower bound.                                             */
85.       for  (status = fread(&next_prod, sizeof (product_t), 1, databasep);
86.             status == 1  &&  params.low_stock > next_prod.stock_num;
87.             status = fread(&next_prod, sizeof (product_t), 1, databasep)) {}
88.
89.       /*  Displays a list of the products that satisfy the search
90.           parameters                                                        */
91.       printf("\nProducts satisfying the search parameters:\n");
92.       while (next_prod.stock_num <= params.high_stock  &&
93.              status == 1) {
94.           if (match(next_prod, params)) {
95.               no_matches = 0;
96.               show(next_prod);
97.           }
98.           status = fread(&next_prod, sizeof (product_t), 1, databasep);
99.       }
100.
101.      /*  Displays a message if no products found                          */
102.      if (no_matches)
103.          printf("Sorry, no products available\n");
104. }
```


34

Thanks for listening!

35