FRESHERS NOW

Texas Instruments Technical Questions and Answers

Q1. How do I write code to retrieve current date and time from the system and display it as a string?

ANS:

Use time( ) function to get current date and time and then ctime( ) function to display it as a string. This is shown in following code snippet.

```
#include
void main( )
{
time_t curtime ;
char ctm[50] ;
time ( &curtime ) ; //retrieves current time &
stores in curtime
printf ( Current Date & Time: %s, ctime (
&curtime ) ) ;
}
```

Q2. How do I write code to find an amount of free disk space available on current drive?

ANS:

Use getdfree( ) function as shown in follow code.

```
#include
#include
#include
#include
main( )
{
int dr ; struct dfree disk ;
long freesp ;
dr = getdisk( ) ;
getdfree ( dr + 1 , &disk ) ;
if ( disk.df_sclus == 0xFFFF )
```

```
{
printf ( getdfree( ) function failed );
exit ( 1 ) ;
}
freesp = ( long ) disk.df_avail
* ( long ) disk.df_bsec
* ( long ) disk.df_sclus ;
printf ( The current drive %c: has %ld bytes
available as free space , A + dr, freesp ) ;
}
```

Q3. How do I write a user-defined function, which deletes each character in a string str1, which matches any character in string str2?

ANS:

The function is as shown below:

```
Compress ( char str1[], char str2[] )
{
int i, j, k ;
for ( i = k = 0 ; str1[i] != ?? ; i++ )
{
for ( j = 0 ; str2[j] != ?? && str2[j] !=
str1[i] ; j++ );
if ( str2[j] == ?? )
str1[k++] = str1[I] ;
}
str1[k] = ??
}
```

Q4. What is the use of randomize( ) and srand( ) function?

ANS:

While generating random numbers in a program, sometimes we require to control the series of numbers that random number generator creates. The process of assigning the random number generators starting number is called seeding the generator. The

randomize( ) and srand( ) functions are used to seed the random number generators. The randomize( ) function uses PCs clock to produce a random seed, whereas the srand( ) function allows us to specify the random number generators starting value.

Q5. How does a C program come to know about command line arguments?

ANS:

When we execute our C program, operating system loads the program into memory. In case of DOS, it first loads 256 bytes into memory, called program segment prefix. This contains file table, environment segment, and command line information. When we compile the C program the compiler inserts additional code that parses the command, assigning it to the argv array, making the arguments easily accessible within our C program.

Q6. The sizeof( ) function does not return the size of the block of memory pointed to by a pointer. Why?

ANS:

The sizeof( ) operator does not know that malloc( ) has been used to allocate a pointer. sizeof( ) gives us the size of pointer itself. There is no handy way to find out the size of a block allocated by malloc( ).

Q7. What is environment and how do I get environment for a specific entry?

ANS:

While working in DOS, it stores information in a memory region called environment. In this region we can place configuration settings such as command path, system prompt, etc. Sometimes in a program we need to access the information contained in environment. The function getenv( ) can be used when we want to access environment for a specific entry. Following program demonstrates the use of this function.

```
#include
#include
main( )
{
```

```
char *path = NULL ;

path = getenv ( PATH ) ;
if ( *path != NULL )
printf ( Path: %s, path ) ;
else
printf ( Path is not set ) ;
}
```

Q8. How do I write printf( ) so that the width of a field can be specified at runtime?

ANS:

This is shown in following code snippet.

```
main( )
{
int w, no ;
printf ( Enter number and the width for the
number field: ) ;
scanf ( %d%d, &no, &w ) ;
printf ( %*d, w, no ) ;
}
```

Here, an * in the format specifier in printf( ) indicates that an int value from the argument list should be used for the field width.

Q9. How do I convert a floating-point number to a string?

ANS:

Use function gcvt( ) to convert a floating-point number to a string. Following program demonstrates the use of this function.

```
#include
main( )
{
char str[25] ;
```

```
float no ;
int dg = 5 ; /* significant digits */
no = 14.3216 ;
gcvt ( no, dg, str ) ;
printf ( String: %s , str ) ;
}
```

Q10. How do I use the function ldexp( ) in a program?

ANS:

The math function ldexp( ) is used while solving the complex mathematical equations. This function takes two arguments, a double value and an int respectively. The order in which ldexp( ) function performs calculations is (n * pow (2, exp)) where n is the double value and exp is the integer. The following program demonstrates the use of this function.

```
#include
#include
void main( )
{
double ans ;
double n = 4 ;
ans = ldexp ( n, 2 ) ;
printf ( The ldexp value is : %lf , ans ) ;
}
```

Here, ldexp( ) function would get expanded as (4 * 2 * 2), and the output would be the ldexp value is : 16.000000

Q11. What are memory models?

ANS:

The compiler uses a memory model to determine how much memory is allocated to the program. The PC divides memory into blocks called segments of size 64 KB. Usually, program uses one segment for code and a second segment for data. A memory model defines the number of segments the compiler can use for each. It is important to know

which memory model can be used for a program. If we use wrong memory model, the program might not have enough memory to execute. The problem can be solved using larger memory model. However, larger the memory model, slower is your program execution. So we must choose the smallest memory model that satisfies our program needs. Most of the compilers support memory models like tiny, small, medium, compact, large and huge.

Q12. Are the expressions *ptr++ and ++*ptr same?

ANS:

No. *ptr++ increments the pointer and not the value pointed by it, whereas ++*ptr increments the value being pointed to by ptr.

Q13. How do I use swab( ) in my program ?

ANS:

The function swab( ) swaps the adjacent bytes of memory. It copies the bytes from source string to the target string, provided that the number of characters in the source string is even. While copying, it swaps the bytes which are then assigned to the target string.

```
#include
#include
#include
main ( )
{
char *str1 = hS eesll snsiasl not eh es as oher  ;
char *str2 ;
clrscr( ) ;
swab ( str1, str2, strlen ( str1 ) ) ;
printf ( The target string is : %s , str2 ) ; // output -- She sells
snails on the sea shore
getch( ) ;
}
```

FRESHERS NOW

Q14. What will be the output of the following program?

```
main( )
{
unsigned int num ;
int i ;
printf ( Enter any number ) ;
scanf ( %u, &num ) ;
for ( i = 0 ; i < 16 ; i++ )
printf ( %d, ( num << i & 1 << 15 ) ? 1 : 0 ) ;
}
```

ANS:

The output of this program is the binary equivalent of the given number. We have used bitwise operators to get the binary number.

Q15. How to get the memory size?

ANS:

Consider the following program

```
#include
#include
main( )
{
int memsize;
memsize = biosmemory( ) ;
printf ( RAM size = %dK ,memsize ) ;
return 0 ;
}
```

The function bios memory uses BIOS interrupt 0x12 to return the size of memory.

Q16. What is garbage collection?

ANS:

Suppose some memory space becomes reusable because a node is released from a linked list. Hence, we want the space to be available for future use. One way to bring this about is to immediately reinsert the space into the free-storage list. However, this method may be too time-consuming for the operating system. The operating system may periodically collect all the deleted space onto the free-storage list. The technique that does this collection is called Garbage Collection. Garbage Collection usually takes place in two steps: First the Garbage Collector runs through all lists, tagging whose cells are currently in use, and then it runs through the memory, collecting all untagged space onto the free-storage list. The Garbage Collection may take place when there is only some minimum amount of space or no space at all left in the free-storage list, or when the CPU is idle and has time to do the collection. Generally speaking, the Garbage Collection is invisible to the programmer.

Q17. How do I get the time elapsed between two function calls?

ANS:

The function difftime( ) finds the difference between two times. It calculates the elapsed time in seconds and returns the difference between two times as a double value.

```
#include
#include
#include
main( )
{
int a[] = { 2, -34, 56, 78, 112, 33, -7, 11, 45, 29, 6 } ;
int s ;
time_t t1, t2 ; // time_t defines the value used for time function
s = sizeof ( a ) / 2 ;
t1 = time ( NULL ) ;
sel_sort ( a, s ) ; // sort array by selection sort
bub_sort ( a, s ) ; // sort array by bubble sort method
t2 = time ( NULL ) ;
printf ( The difference between two function calls is %f, difftime (
t2, t1 ) ) ;
}
```

In the above program we have called difftime( ) function that returns the time elapsed from t1 to t2.

Q18. What is a priority queue?

ANS:

In a stack, the latest element is deleted and in a queue the oldest element is deleted. It may be required to delete an element with the highest priority in the given set of values and not only the oldest or the newest one. A data structure that supports efficient insertions of a new element and deletions of elements with the highest priority is known as priority queue. There are two types of priority queues: an ascending priority queue is a collection of items into which items can be inserted arbitrarily and from which only the smallest item can be removed. A descending order priority queue is similar but allows only the largest item to be deleted.

Q19. What is the difference between a null pointer, a NULL macro, the ASCII NUL character and a null string?

ANS:

A null pointer is a pointer which does not point anywhere. A NULL macro is used to represent the null pointer in source code. It has a value 0 associated with it. The ASCII NUL character has all its bits as 0 but does not have any relationship with the null pointer. The null string is just another name for an empty string.

Q20. What does the error Null Pointer Assignment mean and what causes this error?

ANS:

The Null Pointer Assignment error is generated only in small and medium memory models. This error occurs in programs which attempt to change the bottom of the data segment. In Borlands C or C++ compilers, Borland places four zero bytes at the bottom of the data segment, followed by the Borland copyright notice Borland C++ - Copyright 1991 Borland Intl.. In the small and medium memory models, a null pointer points to DS:0000.

Thus assigning a value to the memory referenced by this pointer will overwrite the first zero byte in the data segment. At program termination, the four zeros and the copyright banner are checked. If either has been modified, then the Null Pointer Assignment error is generated.

Q21. How to build an expression trees?

ANS:

An expression tree is a binary tree which is built from simple operands and operators of an (arithmetic or logical) expression by placing simple operands as the leaves of a binary tree and the operators as the interior nodes. If an operator is binary, then it has two nonempty subtrees, that are its left and right operands (either simple operands or sub expressions).

If an operator is unary, then only one of its subtrees is nonempty, the one on the left or right according as the operator is written on the right or left of its operand. We traditionally write some unary operators to the left of their operands, such as - (unary negation) or the standard functions like log( ), sin( ) etc. Others are written on the right, such as the factorial function ()!. If the operator is written on the left, then in the expression tree we take its left subtree as empty. If it appears on the right, then its right subtree will be empty.

Q22. Can we get the remainder of a floating point division?

ANS:

Yes. Although the % operator fails to work on float numbers we can still get the remainder of floating point division by using a function fmod( ). The fmod( ) function divides the two float numbers passed to it as parameters and returns the remainder as a floating-point value. Following program shows fmod( ) function at work.

```
#include
main( )
{
printf ( %f, fmod ( 5.15, 3.0 ) ) ;
}
```

The above code snippet would give the output as 2.150000.

Q23. How to extract the integer part and a fractional part of a floating point number?

ANS:

C function modf( ) can be used to get the integer and fractional part of a floating point.

```
#include math.h
main( )
{
double val, i, f ;
val = 5.15 ;
f = modf ( val, &i ) ;
printf ( For the value %f integer part = %f and fractional part = %f,
val, i, f ) ;
}
```

The output of the above program will be:
For the value 5.150000 integer part = 5.000000 and fractional part = 0.150000

Q24. What is wrong with the following declaration: char* ptr1, ptr2 ; get errors when I try to use ptr2 as a pointer.

ANS:

char * applies only to ptr1 and not to ptr2. Hence ptr1 is getting declared as a char pointer, whereas, ptr2 is being declared merely as a char. This can be rectified in two ways:

```
char *ptr1, *ptr2 ;
typedef char* CHARPTR ; CHARPTR ptr1, ptr2 ;
```

Q25. Why the output of sizeof (a) is 2 and not 1 ?

ANS:

Character constants in C are of type int, hence sizeof (a) is equivalent to sizeof (int), i.e. 2. Hence the output comes out to be 2 bytes.

Q26. Can we use scanf( ) function to scan a multiple words string through keyboard?

ANS:

Yes. Although we usually use scanf( ) function to receive a single word string and gets( ) to receive a multi-word string from keyboard we can also use scanf( ) function for scanning a multi-word string from keyboard. Following program shows how to achieve this.

```
main( )
{
char buff[15] ;
scanf ( %[^ ]s, buff ) ;
puts ( buff ) ;
}
```

In the scanf( ) function we can specify the delimiter in brackets after the ^ character. We have specified as the delimiter. Hence scanf( ) terminates only when the user hits Enter key.

Q27. How to set the system date through a C program?

ANS:

We can set the system date using the setdate( ) function as shown in the following program. The function assigns the current time to a structure date.

```
#include stdio.h
#include dos.h
main( )
{
struct date new_date ;
new_date.da_mon = 10 ;
```

```
new_date.da_day = 14 ;
new_date.da_year = 1993 ;
setdate ( &new_date ) ;}
```

Q28. What is a heap?

ANS:

Heap is a chunk of memory. When in a program memory is allocated dynamically, the C run-time library gets the memory from a collection of unused memory called the heap. The heap resides in a programs data segment.

Therefore, the amount of heap space available to the program is fixed, and can vary from one program to another.

Q29. How to obtain a path of the given file?

ANS:

The function searchpath( ) searches for the specified file in the subdirectories of the current path. Following program shows how to make use of the searchpath( ) function.

```
#include dir.h
void main ( int argc, char *argv[] )
{
char *path ;
if ( path = searchpath ( argv[ 1 ] ) )
printf ( Pathname : %s , path ) ;
else
printf ( File not found ) ;
}
```

Q30. How to write a swap( ) function which swaps the values of the variables using bitwise operators?

ANS:

Here is the swap( ) function.

```
swap ( int *x, int *y )
{
*x ^= *y ;
*y ^= *x ;
*x ^= *y ;
}
```

The swap( ) function uses the bitwise XOR operator and does not require any temporary variable for swapping.

Q31. On including a file twice I get errors reporting redefinition of function. How can I avoid duplicate inclusion?

ANS:

Redefinition errors can be avoided by using the following macro definition. Include this definition in the header file.

```
#if !defined filename_h
#define filename_h
/* function definitions */
#endif
```

Replace filename_h with the actual header file name. For example, if name of file to be included is goto.h then replace filename_h with goto_h.

Q32. What is the output of the program?

```
void main ()
{ int i = 0 , a[3] ;
a[i] = i++;
printf (?%d,a[i]) ;
}
```

ANS:

The output for the above code would be a garbage value.

Q33. Why does not the following code give the desired result?

```
int x = 3000, y = 2000 ;
long int z = x * y ;
```

ANS:

Here the multiplication is carried out between two ints x and y, and the result that would overflow would be truncated before being assigned to the variable z of type long int. However, to get the correct output, we should use an explicit cast to force long arithmetic as shown below:

```
long int z = ( long int ) x * y ;
```

Note that (long int)(x * y) would not give the desired effect.

Q34. How do I know how many elements an array can hold?

ANS:

The amount of memory an array can consume depends on the data type of an array. In DOS environment, the amount of memory an array can consume depends on the current memory model (i.e. Tiny, Small, Large, Huge, etc.). In general an array cannot consume more than 64 kb. Consider following program, which shows the maximum number of elements an array of type int, float and char can have in case of Small memory model.

```
main( )
{
int i[32767] ;
float f[16383] ;
char s[65535] ;
}
```

Q35. Why does not the following statement work?

```
char str[ ] = Hello ;
strcat ( str, ! ) ;
```

ANS:

The string function strcat( ) concatenates strings and not a character. The basic difference between a string and a character is that a string is a collection of characters, represented by an array of characters whereas a character is a single character. To make the above statement work writes the statement as shown below:

```
strcat (str, !) ;
```

Q36. Is the following code fragment correct?

```
const int x = 10 ;
int arr[x] ;
```

ANS:

No! Here, the variable x is first declared as an int so memory is reserved for it. Then it is qualified by a const qualifier. Hence, const qualified object is not a constant fully. It is an object with read only attribute, and in C, an object associated with memory cannot be used in array dimensions.

Q37. How do I change the type of cursor and hide a cursor?

ANS:

We can change the cursor type by using function _setcursortype( ). This function can change the cursor type to solid cursor and can even hide a cursor. Following code shows how to change the cursor type and hide cursor.

```
#include
main( )
{
/* Hide cursor */
_setcursortype ( _NOCURSOR ) ;
/* Change cursor to a solid cursor */
_setcursortype ( _SOLIDCURSOR ) ;
```

```
/* Change back to the normal cursor */
_setcursortype ( _NORMALCURSOR ) ;
}
```

Q38. How does free( ) know how many bytes to free?

ANS:

The malloc( ) / free( ) implementation remembers the size of each block allocated and returned, so it is not necessary to remind it of the size when freein;

Q39. How do I determine amount of memory currently available for allocating?

ANS:

We can use function coreleft( ) to get the amount of memory available for allocation. However, this function does not give an exact amount of unused memory. If, we are using a small memory model, coreleft( ) returns the amount of unused memory between the top of the heap and stack. If we are using a larger model, this function returns the amount of memory between the highest allocated memory and the end of conventional memory. The function returns amount of memory in terms of bytes.

Q40. What is a stack?

ANS:

The stack is a region of memory within which our programs temporarily store data as they execute. For example, when a program passes parameters to functions, C places the parameters on the stack. When the function completes, C removes the items from the stack. Similarly, when a function declares local variables, C stores the variables values on the stack during the functions execution. Depending on the programs use of functions and parameters, the amount of stack space that a program requires will differ.

Q41. How to distinguish between a binary tree and a tree?

ANS:

A node in a tree can have any number of branches. While a binary tree is a tree structure in which any node can have at most two branches. For binary trees we distinguish between the subtree on the left and subtree on the right, whereas for trees the order of the subtrees is irrelevant.

Q42. Can we get the mantissa and exponent form of a given number?

ANS:

The function frexp( ) splits the given number into a mantissa and exponent form. The function takes two arguments, the number to be converted as a double value and an int to store the exponent form. The function returns the mantissa part as a double value.

Q43. How does C compiler store elements in a multi-dimensional array?

ANS:

The compiler maps multi-dimensional arrays in two ways: Row major order and Column order. When the compiler places elements in columns of an array first then it is called column-major order. When the compiler places elements in rows of an array first then it is called row-major order. C compilers store multidimensional arrays in row-major order. For example, if there is a multi-dimensional array a[2][3], then according row-major order, the elements would get stored in memory following order:

a[0][0], a[0][1], a[0][2], a[1][0], a[1][1], a[1][2]

Q44. What is the difference between these two declarations?

struct str1 { ... } ;
typedef struct { ... } str2 ;

ANS:

The first form declares a structure tag whereas the second declares a typedef. The main difference is that the second declaration is of a slightly more abstract type -- its users do not necessarily know that it is a structure, and the keyword struct is not used when declaring instances of it.

Q45. When we open a file, how does functions like fread( )/fwrite ( ), etc. get to know from where to read or to write the data?

ANS:

When we open a file for read/write operation using function like fopen(), it returns a pointer to the structure of type FILE. This structure stores the file pointer called position pointer, which keeps track of current location within the file. On opening file for read/write operation, the file pointer is set to the start of the file. Each time we read/write a character, the position pointer advances one character. If we read one line of text at a step from the file, then file pointer advances to the start of the next line.

If the file is opened in append mode, the file pointer is placed at the very end of the file. Using fseek( ) function we can set the file pointer to some other place within the file.

Q46. How do I print a floating-point number with higher precision say 23.34568734 with only precision up to two decimal places?

ANS:

This can be achieved through the use of suppression char * in the format string of printf( ) as shown in the following program.

```
main( )
{
int i = 2 ;
float f = 23.34568734 ;
printf ( %.*f, i, f ) ;
}
```

The output of the above program would be 23.35.

Q47. How do I print the contents of environment variables?

ANS:

The following program shows how to achieve this:

```
main( int argc, char *argv[ ], char *env[ ] )
{
int i = 0 ;
clrscr( ) ;
while ( env[ i ] )
printf ( %s, env[ i++ ] ) ;
}
```

main( ) has the third command line argument env, which is an array of pointers to the strings. Each pointer points to an environment variable from the list of environment variables.

Q48. How to obtain the current drive through C?

ANS:

We can use the function _getdrive( ) to obtain the current drive. The _getdrive ( ) function uses DOS function 0X19 to get the current drive number

```
#include
main( )
{
int disk ;
disk = _getdrive( ) + A - 1 ;
printf ( The current drive is: %c , disk ) ;
}
```

Q49. Can we get the x and y coordinate of the current cursor position?

ANS:

The function wherex( ) and wherey( ) returns the x-coordinate and y-coordinate of the current cursor position respectively. Both the functions return an integer value. The value returned by wherex( ) is the horizontal position of cursor and the value returned by wherey( ) is the vertical position of the cursor.

Q50. Why is it not possible to scan strings from keyboard in case of array of pointers to string?

ANS:

When an array is declared, dimension of array should be specified so that compiler can allocate memory for the array. When array of pointers to strings is declared its elements would contain garbage addresses. These addresses would be passed to scanf( ). So strings can be received but they would get stored at unknown locations. This is unsafe.