OSW QUIZ SOLUTION NOVEMBER-2023

Operating System Workshop (CSE 3541)

Programme: B.Tech. (CSE - N) | **Semester: 5th**
Full marks: 30 | **Quiz no.: 03**
Date: 27/11/2023 | **Time: 02 Hrs**

NB : All codes must be written using C language.

---

1. (a) Write a C function to print the binary equivalent of a decimal number using recursion.

ANS :

```
void binary_print(int num){
        if(num != 0)
        {
                int r = num % 2;
                binary_print(num/2);
                printf("%d ", r);
        }
}
```

(b) Find the output of the following code segment and discuss the reason behind the obtained output.

```
#include<stdio.h>
int main(){
        float a=1.2;
        if(a==1.2) printf("C\n");
        else printf("U\n");
        return(0)
}
```

Ans : Output : U

Reason : In `a==1.2` LHS is float and RHS is double which are different number of bytes. Floating point numbers are stored in computers as binary sequences divided into different fields, one field storing the mantissa, the other the exponent. Based on the size of the data type the precision in mantissa part changes. Hence, the conditional checking fails and `U` gets printed.

(c) Write a C function to concatenate two strings without using any inbuilt concatenation functions (to allocate memory use malloc() funtion).

ANS :

```
char * binary_print(char *str1,char *str2){
        int len_str1, len_str2;
        for(len_str1=0; str1[len_str1]!='\0'; len_str1++);
        for(len_str2=0; str2[len_str2]!='\0'; len_str2++);
        char *str3 = (char *) malloc(len_str1+len_str2+1);
        for(int i=0; str1[i]!='\0'; i++) str3[i] = str1[i];
        for(int i=0; str2[i]!='\0'; i++) str3[i+len_str1] = str2[i];
        str3[len_str1+len_str2] = '\0';
        return(str3);
}
```

2. (a) Write a C program to find the sum of two floating point numbers given in command line argument.

ANS :

```
#include<stdio.h>
int main(int argc, char **argv)
{
        float f1, f2;
        sscanf(argv[1], "%f", &f1);
        sscanf(argv[2], "%f", &f2);
        printf("%f + %f = %f\n", f1 , f2, f1+f2);
        return(0);
}
```

(b) **Fill in the blanks so that the final out put will be "01":**

```
#include<stdio.h>
#include<string.h>
int main(){
        char str[20]="Dibya;01-12-1990";
        char *rem, *ptr;
        ptr = strtok_r(_____ , _____ , _____ );
        ptr = strtok( _____ , _____ );
        printf("%s\n",ptr);
        return(0);
}
```

ANS :
```
ptr = strtok_r(str , ";" , &rem );
ptr = strtok( rem , "-" );
```

(c) **Write a C function to find reverse of a string using pointers and without using any inbuilt function.**

ANS :
```
void str_rev(char *str)
{
        int len;char temp ;
        for(len=0;str[len]!='\0';len++);
        for(int i = 0; i<=len/2 ; i++)
        {
                temp = str[len-1-i];
                str[len-1-i] = str[i] ;
                str[i] = temp;
        }
}
```

3. (a) **Find the output of the following code segment. If there is any runtime or compilation error, then mention the exact reason of the error.**

```
#include<stdio.h>
#define SWAP(a, b, c) {c t; t=a; a=b; b=t;}
int main()
{
        int a=12, b=15;
        float c=2.0, d=3.0;
        SWAP(a,b, int);
        SWAP(c,d, float);
        printf("a=%d; b=%d\n",a,b);
        printf("c=%f; d=%f\n",c,d);
        return(0);
}
```

ANS :
```
a=15; b=12
c=3.000000; d=2.000000
```

(b) **Write a C program to tokenize the date in given form "dd-mm-yyyy" and print the month name in words.**

ANS :
```
#include<stdio.h>
int main()
{
        int d, m, y;
        char month[12][4]={"JAN", "FEB", "MAR", "APR", "MAY", "JUN",
                                "JUL", "AUG", "SEP", "OCT", "NOV", "DEC"};
        scanf("%d-%d-%d", &d, &m, &y);
        printf("Month is : %s\n", month[m-1]);
        return(0);
}
```

(c) **Write a C program to remove the duplicate numbers from an array.**

ANS :

```
#include<stdio.h>
int main(){
  int arr[] = {1,2,3,1,1,2,3,4,5};
  int len = sizeof(arr)/sizeof(int);
  for(int i=0; i<len; i++){
    for(int j=i+1; j<len; ){
      if(arr[i] == arr[j]){
        for(int k=j; k<len-1; arr[k]=arr[k+1],k++);
        len=len-1;
      }
      else{
        j=j+1;
      }
    }
  }
  for(int i=0; i<len; printf("%d ",arr[i]),i++);
  printf("\n");
  return(0);
}
```

4. (a) What is the difference between process and program? Write a short note on fork system call.

ANS : A program is a set of instructions and data stored on a disk, while a process is an instance of a program in execution, loaded into memory.

fork() system call

```
#include <unistd.h>
pid_t fork(void);
```

- **A process can create a new process by calling fork.**
- **The calling process becomes the parent, and the created process is called the child.**
- **If fork() returns a negative value, the creation of a child process was unsuccessful.**
- **fork() returns a zero to the newly created child process.**
- **fork() returns a positive value, the process ID of the child process, to the parent.**

(b) Draw a neat diagram for the layout of process image. With an example show how the uninitialized and initialized static data affect the executable size.

ANS :

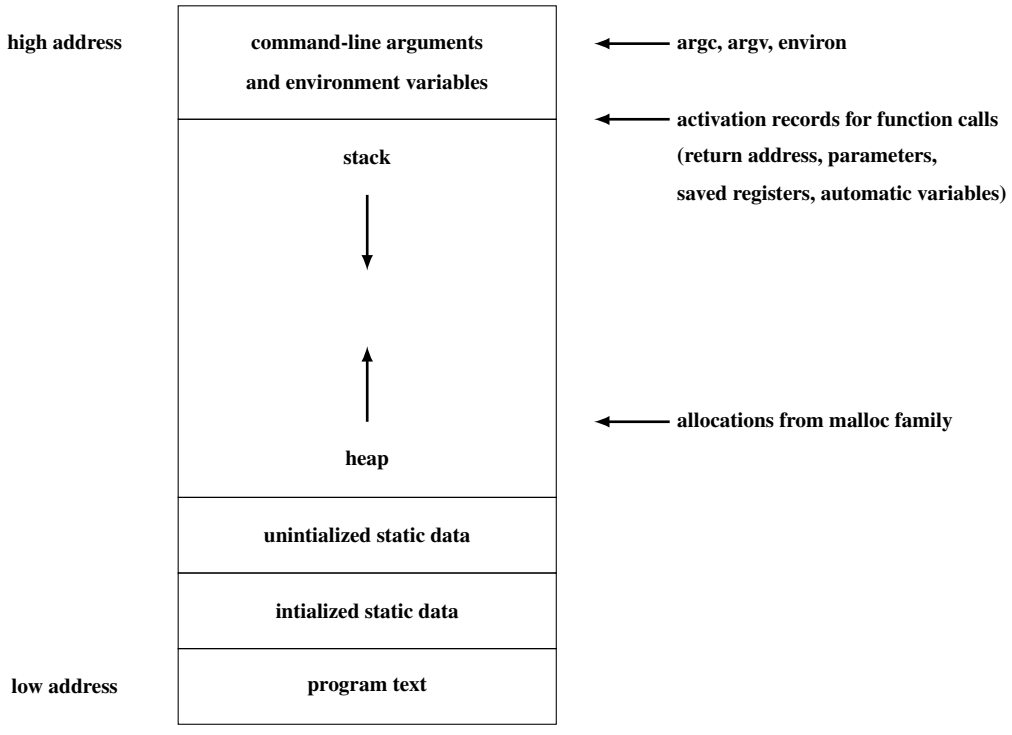


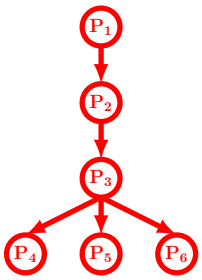
Figure 1: **Process Image**

```
Program 1:
-----------
#include<stdio.h>
int x[10000]; // uninitialized static variable
int main(){
return(0);
}
```

```
Program 2:
-----------
#include<stdio.h>
int x[10000]={0}; // initialized static variable
int main(){
        return(0);
}
```

Initialized static data is part of the executable module hence the object file size of program 2 will more than that of the program 1.

(c) Write a code to generate the following correlated processes where arrow indicate parent child relation. Make sure no zombie process will not be created.



ANS :

```
#include<stdio.h>
#include<unistd.h>
int main(){
        if(fork()==0){
                if(fork()==0){
                        for(int i=0;i<3;i++){
                                if(fork()==0)
                                        break;
                        }
                }
                else{
                        wait(NULL);
                }
        }
        else{
                wait(NULL);
        }
        return(0);
}
```

5. (a) What is a zombie process? Write a C code to create a zombie process. Modify the same code to avoid the creation of zombie process.

ANS : Zombie process is a process that has completed its execution but still has an entry in the process table. It exists in a "zombie" state, retaining its process ID (PID) and entry in the process table until the parent process retrieves its exit status.

```
#include<stdio.h>
#include<unistd.h>
int main(){
        if(fork()!=0)
                sleep(100);
        return(0);
}
```

```
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
int main(){
        if(fork()!=0)
                sleep(100);
        else
                wait(NULL);
        return(0);
}
```

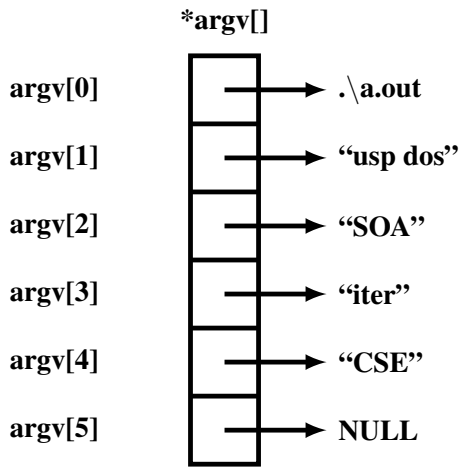
(b) What is an orphan process. Explain with a example code.

ANS : An orphan process is a process whose parent process has finished or terminated unexpectedly before the orphan process completes its execution. Orphan processes are then adopted by the operating system's init process, which ensures their continued execution and handles their termination, preventing them from becoming zombie processes.

```
#include<stdio.h>
#include<unistd.h>
int main(){
        if(fork()==0)
                sleep(100);
        return(0);
}
```

(c) Draw the diagram of the argument array prepared by the shell given to
`int main(int argc, char *argv[])`
for the command line
`$./a.out ``usp dos" SOA iter CSE.`

ANS :



— X —