

**Department of Computer Science & Engineering**

**Operating Systems - UE20CS254**

**LAB ASSIGNMENT PROGRAMS AND OUTPUTS**

|  |  |
| --- | --- |
| SL No: | Programs and outputs |
|  | Write program to create a child process which lists all files in the current directory along with the size (Avoid creation of Zombie process).  Code:   |  | | --- | | #include<stdio.h>  #include<stdlib.h>  #include<unistd.h>  #include<sys/wait.h>  int main(){    pid\_t p1,p2;    int y=0;    p1=fork();      if(p1>0){      wait(NULL);      y=y-1;      printf("This is parent. Process Id=%d,y=%d\n",getpid(),y);      execl("/bin/ls","home/Desktop",NULL);    }    else if(p1==0){    y++;    printf("This is child.Process Id = %d,Parent process =%d,y=%d\n",getppid(),getpid(),y);       char \*a[]={NULL};     execv("/home/Desktop/gcc.c",a);     exit(0);     }     else{     printf("fork creation failed\n");     }  } |   Screenshot of the output:   |  | | --- | |  | |
|  | Given an array, use fork and pipes to create two processes. Each process will find the sum of first half and second half of the elements of an array and return that sum through a pipe to the parent process.  Code:   |  | | --- | | #include<stdio.h>  #include<string.h>  #include<unistd.h>  #include<sys/types.h>  #include<sys/wait.h>  int main(int argc, char\* argv[]) {      int fd[2],n=8;      if (pipe(fd) == -1) {          return 1;      }        int pid = fork();      if (pid == -1) {          return 2;      }        if (pid == 0) {          // Child  process          close(fd[0]);              int arr[8] = {9,4,34,1,8,3,7,2};          printf("The elements are : ");          for(int i=0;i<8;i++)              printf("%d ",arr[i]);            printf("\n");            //          if (write(fd[1], &n, sizeof(int)) < 0) {              return 3;          }          printf("Sent n = %d\n", n);            if (write(fd[1], &arr, sizeof(int) \* n) < 0) {              return 4;          }            //printf("Sent array\n");          close(fd[1]);      } else {          close(fd[1]);          int arr[10];          int n, i, sum = 0,sum1=0;            if (read(fd[0], &n, sizeof(int)) < 0) {              return 5;          }          printf("Received n = %d\n", n);          if (read(fd[0], &arr, sizeof(int) \* n) < 0) {              return 6;          }          //printf("Received array\n");            close(fd[0]);          for (i = 0; i < n/2; i++) {              sum += arr[i];          }          for (i = n/2; i < n; i++) {              sum1 += arr[i];          }          printf("Result is %d,%d\n", sum,sum1);          wait(NULL);      }        return 0;  } |   Screenshot of the output:   |  | | --- | |  | |
|  | Write a C programme to simulate Segmentation:  Take as Input:   1. Segmentation number 2. Base Address 3. Segment limit   Code:   |  | | --- | | #include <stdio.h>  #include <stdlib.h>  struct list  {      int seg;      int base;      int limit;      struct list \*next;  } \* p;  void insert(struct list \*q, int base, int limit, int seg)  {      if (p == NULL)      {          p = malloc(sizeof(struct list));          p->limit = limit;          p->base = base;          p->seg = seg;          p->next = NULL;      }      else      {          while (q->next != NULL)          {              q = q->next;              printf("yes");          }          q->next = malloc(sizeof(p));          q->next->limit = limit;          q->next->base = base;          q->next->seg = seg;          q->next->next = NULL;      }  }  int find(struct list \*q, int seg)  {      while (q->seg != seg)      {          q = q->next;      }      return q->limit;  }  int search(struct list \*q, int seg)  {      while (q->seg != seg)      {          q = q->next;      }      return q->base;  }  int main()  {      p = NULL;      int seg, offset, limit, base, c, s, physical;      printf("Enter segment table: \n");      printf("Enter -1 as segment value for termination\n");      do      {          printf("Enter segment number: ");          scanf("%d", &seg);          if (seg != -1)          {              printf("Enter base value : ");              scanf("%d", &base);              printf("Enter value for limit : ");              scanf("%d", &limit);              insert(p, base, limit, seg);          }      } while (seg != -1);      printf("Enter offset : ");      scanf("%d", &offset);      printf("Enter bsegmentation number : ");      scanf("%d", &seg);      c = find(p, seg);      s = search(p, seg);      if (offset < c)      {          physical = s + offset;          printf("Address in physical memory % d\n", physical);      }      else      {          printf("error");      }  } |   Screenshot of the output:   |  | | --- | |  | |
|  | Write a C program to list all the files that have been created after a certain date.  Inputs to the program:   1. Directory 2. Date   Code:   |  | | --- | | #include<stdio.h>  #include<dirent.h>  #include<sys/types.h>  #include<sys/stat.h>  #include<time.h>  #include<string.h>  #include<stdlib.h>  int main(){  DIR \*dir;  struct dirent \*dirent;  struct stat statbuf;  char path[100];  char date[100];  printf("Enter the path: ");  scanf("%s", path);  printf("Enter the date in dd/mm/yyyy format: ");  scanf("%s", date);  dir = opendir(path);  if(dir == NULL)  {  printf("Error in opening the directory\n");  return 1;  }  while((dirent = readdir(dir)) != NULL)  {  if(stat(dirent->d\_name,&statbuf) == -1)  {  printf("Error in stat\n");  return 1;  }  if(strcmp(date,ctime(&statbuf.st\_ctime)) <= 0)  printf("%s\n", dirent->d\_name);}  closedir(dir);  return 0;  } |   Screenshot of the output:   |  | | --- | |  | |
|  | Write a C programme to change the ownership of files in a directory created after a certain date.  Inputs to the program:   1. Directory 2. Date 3. New permission to be set as run time arguments(access code).   Code:   |  | | --- | | #include<stdio.h>  #include<dirent.h>  #include<sys/types.h>  #include<sys/stat.h>  #include<time.h>  #include<string.h>  #include<stdlib.h>  int main(){  DIR \*dir;  struct dirent \*dirent;  struct stat statbuf;  char path[100];  char date[100];  printf("Enter the path: ");  scanf("%s", path);  printf("Enter the date in dd/mm/yyyy format: ");  scanf("%s", date);  printf("Enter access modes as a 4 digit number:");  int mode = scanf("%d", &mode);  dir = opendir(path);  if(dir == NULL)  {  printf("Error in opening the directory\n");  return 1;  }  while((dirent = readdir(dir)) != NULL)  {  if(stat(dirent->d\_name,&statbuf) == -1)  {  printf("Error in stat\n");  return 1;  }if(strcmp(date,ctime(&statbuf.st\_ctime)) <= 0)  {  printf("%s\n", dirent->d\_name);  chmod(path, mode);  }  }  closedir(dir);  return 0;  } |   Screenshots of the output:   |  | | --- | |  |  |  | | --- | |  | |