DHRUV GARG 16 BCE 1190

OPERATING SYSTEMS

DIGITAL ASSIGNMENT 2

Q1. Discuss the pros and cons of various structures that can be used to implement page tables.

Page table has page table entries where each page table entry stores a frame number and optional status (like protection) bits. Many of status bits used in the virtual memory system. The most important thing in PTE is frame number. Each page table entry has the following information:

1. **Frame Number -** It gives the frame number in which the current page you are looking for is present. The number of bits required depends on the number of frames.

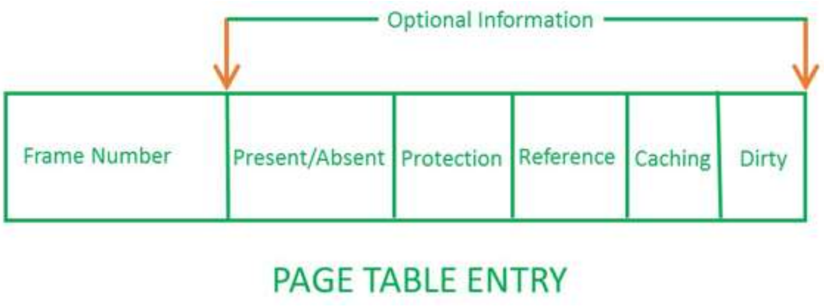
2. **Present/Absent bit –** Present or absent bit tells whether a particular page you are looking for is present or absent. In case if it is not present, that is called Page Fault.

3. **Protection bit –** Protection bit says that what kind of protection you want on that page (read, write etc).

4. **Referenced bit –** Referenced bit will say whether this page has been referred in the last clock cycle or not. It is set to 1 by hardware when the page is accessed.

5. **Caching enabled/disabled –** The main memory contains the latest information which is typed by the user. We want the information to have consistency, which means that whatever information is provided by the user, the CPU should be able to see it as fast as possible. That is the reason we want to disable caching. This bit enable or disable caching of the page.

6. **Modified bit –** Modified bit tells whether the page has been modified or not. It is set to 1 by hardware on write-access to page which is used to avoid writing when swapped out. Sometimes this modified bit is also called as the Dirty bit.



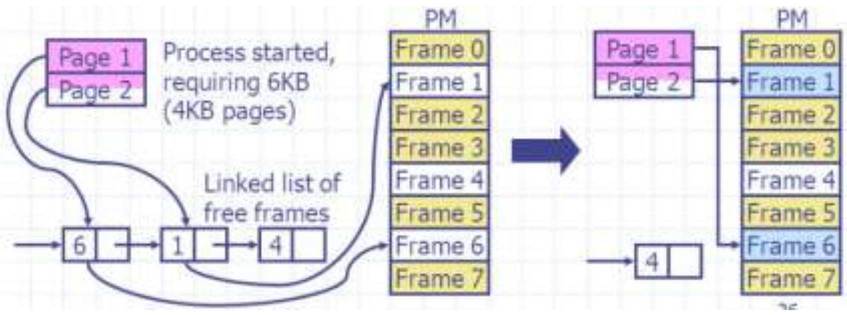
The structures used to implement page tables along with their pros and cons are:

A. NORMAL MEMORY STRUCTURES (ARRAYS, LINKED LISTS)

Advantages

a. Avoids external fragmentation

b. Voids problem of varying sized memory chunks



Disadvantages

a. Consider a 32-bit logical address space as on modern computers

b. Page size of 4 KB (2^{12})

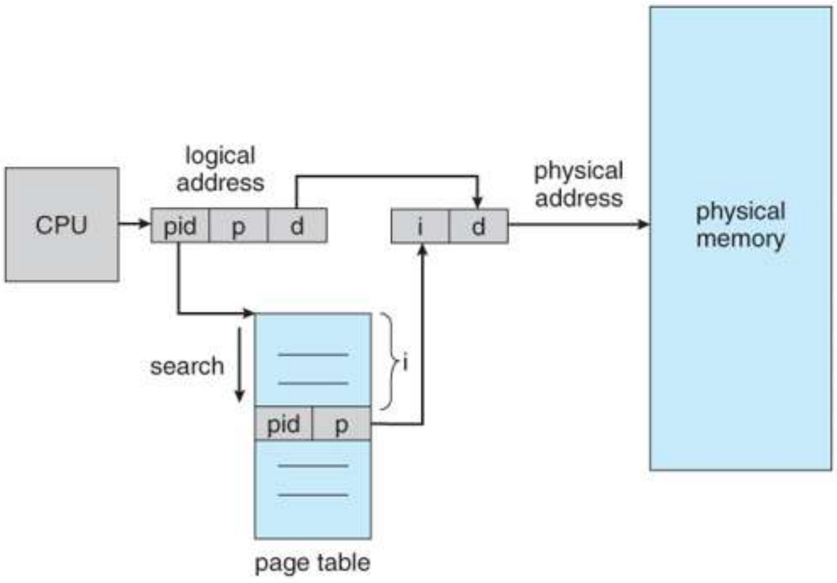
c. Page table would have 1 million entries (2^{32} / 2^{12})

d. If each entry is 4 bytes -> 4 MB of physical address space / memory for page table alone.That amount of memory used to cost a lot

B. INVERTED PAGE TABLE

Advantages

a. Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs



Disadvantages

a. A major problem with this design is poor cache locality caused by the hash function.

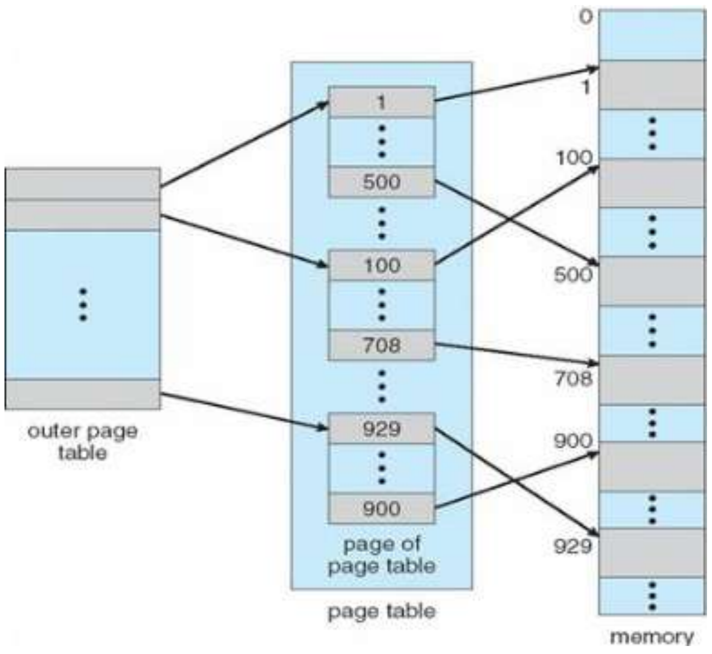
b. It is somewhat slow to remove the page table entries of a given process; the OS may avoid reusing per-process identifier values to delay facing this

C. HIERARCHIAL PAGE TABLES

Advantages

a. We can create smaller 1024-entry 4K pages that cover 4M of virtual memory.

b. This is useful since often the top-most parts and bottom-most parts of virtual memory are used in running a process - the top is often used for text and data segments while the bottom for stack, with free memory in between.



Disadvantages

a. The degree of paging will depend on the page size.

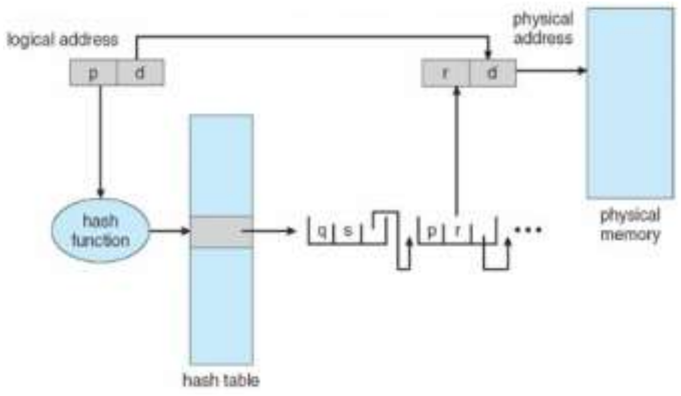
D. Hashed page tables

Advantages

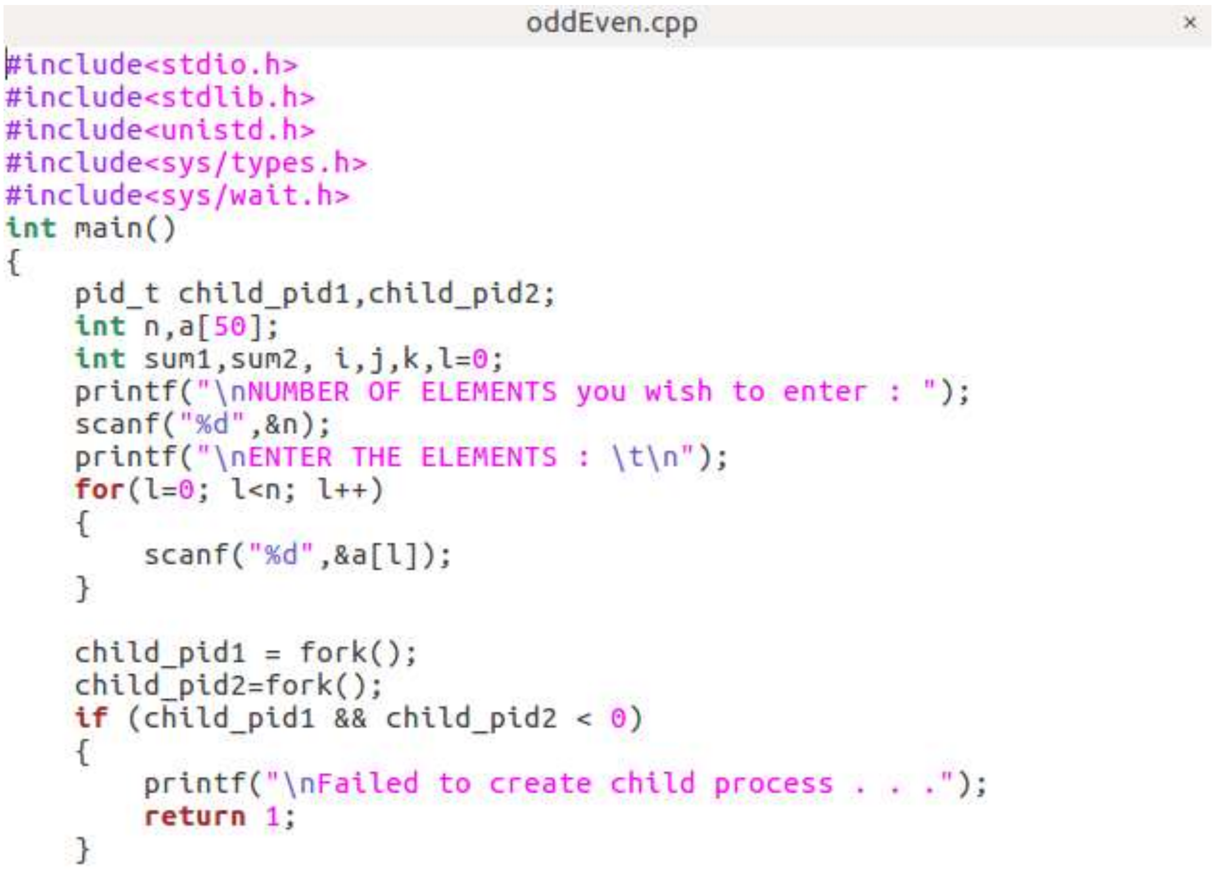
a. For address space larger than 32 bits hashing is the only option

Disadvantages

a. Poor cache locality caused by hash functions



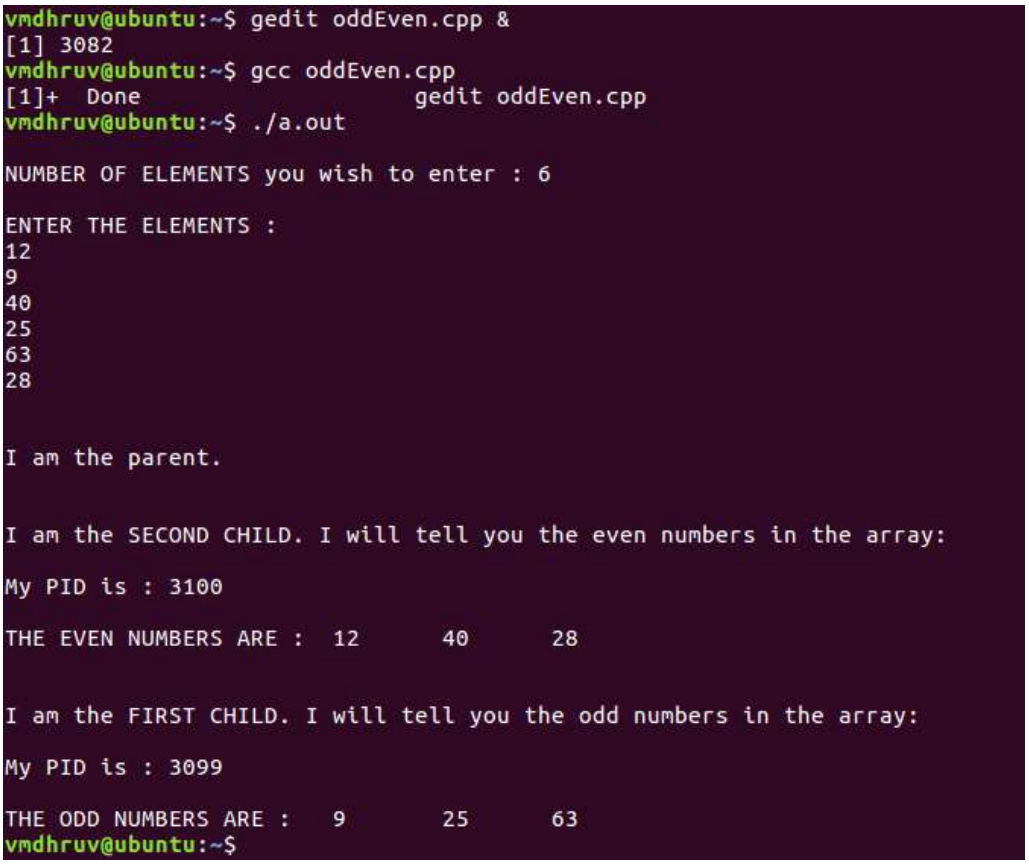
Q2. Develop a C program to find the odd and even number of given n numbers in a array using two child processes (one to print odd and another for even).



oddEven.cpp

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{
    pid_t child_pid1,child_pid2;
    int n,a[50];
    int sum1,sum2, i,j,k,l=0;
    printf("\nNUMBER OF ELEMENTS you wish to enter : ");
    scanf("%d",&n);
    printf("\nENTER THE ELEMENTS : \t\n");
    for(l=0; l<n; l++)
    {
        scanf("%d",&a[l]);
    }

    child_pid1 = fork();
    child_pid2=fork();
    if (child_pid1 && child_pid2 < 0)
    {
        printf("\nFailed to create child process . . .");
        return 1;
    }
if (child_pid1== 0)
{
    printf("\n\nI am the FIRST CHILD. I will tell you the odd numbers in the array:\n");
    printf("\nMy PID is : %d\n",getpid());
    printf("\nTHE ODD NUMBERS ARE : \t");
    for(j=0; j<n; j++)
    {
        if(a[j]%2!=0)
        {
            printf("%d\t",a[j]);
        }
    }
    printf("\n");
}


    else if (child_pid2 == 0)
    {
        printf("\n\nI am the SECOND CHILD. I will tell you the even numbers in the array:\n");
        printf("\nMy PID is : %d\n",getpid());
        printf("\nTHE EVEN NUMBERS ARE : \t");

        for(j=0; j<n; j++)
        {
            if(a[j]%2==0)
            {
                printf("%d\t",a[j]);
            }
        }
        printf("\n");
    }
    else
    {
        printf("\n\nI am the parent.\n");
        wait(&child_pid1);
        wait(&child_pid2);
    }
    return 0;
}
```

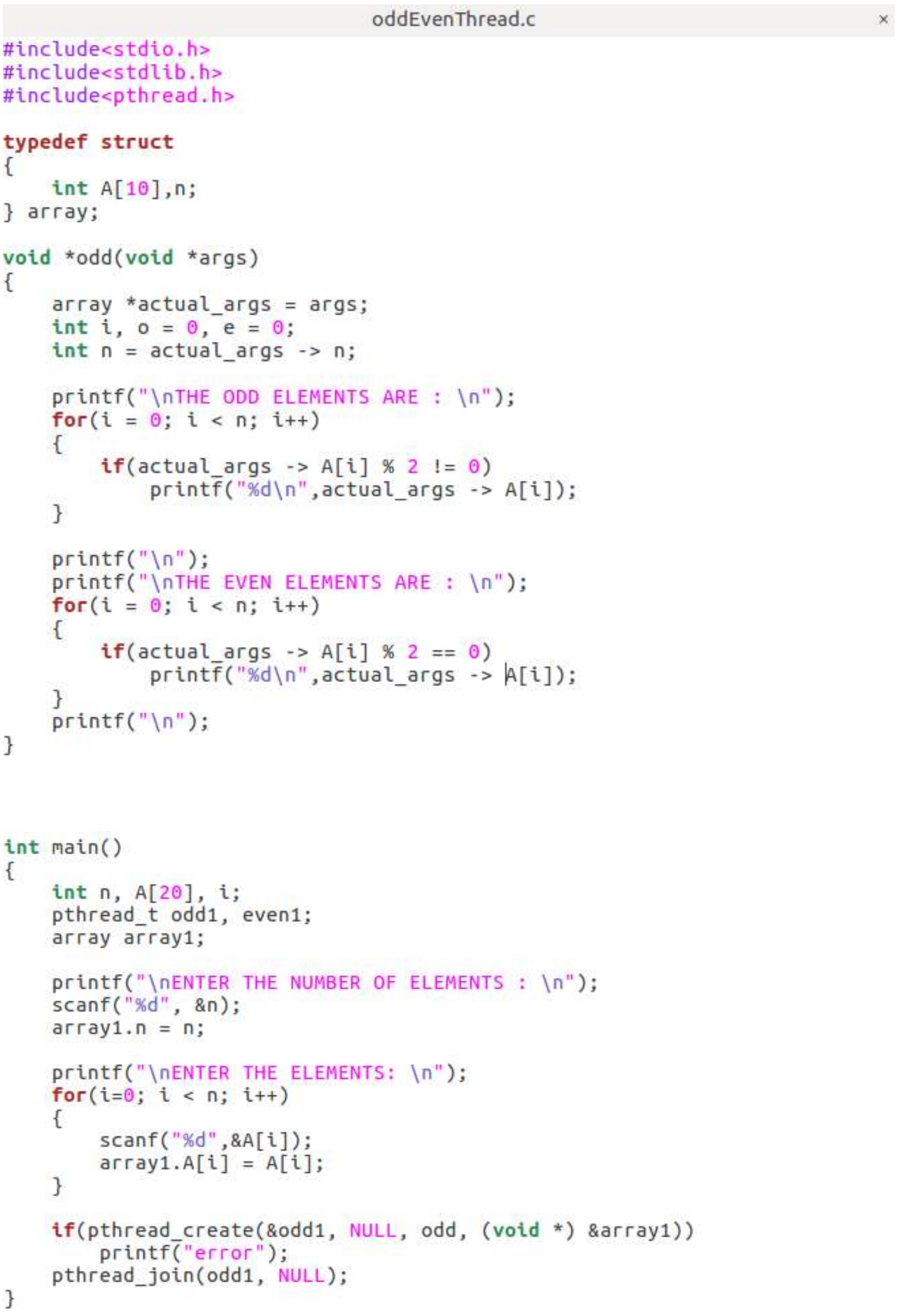
OUTPUT



```
vmdhruv@ubuntu:~$ gedit oddEven.cpp &
[1] 3082
vmdhruv@ubuntu:~$ gcc oddEven.cpp
[1]+  Done                    gedit oddEven.cpp
vmdhruv@ubuntu:~$ ./a.out

NUMBER OF ELEMENTS you wish to enter : 6

ENTER THE ELEMENTS :
12
9
40
25
63
28


I am the parent.


I am the SECOND CHILD. I will tell you the even numbers in the array:

My PID is : 3100

THE EVEN NUMBERS ARE :  12      40      28


I am the FIRST CHILD. I will tell you the odd numbers in the array:

My PID is : 3099

THE ODD NUMBERS ARE :   9       25      63
vmdhruv@ubuntu:~$
```

DHRUV GARG 16 BCE 1190

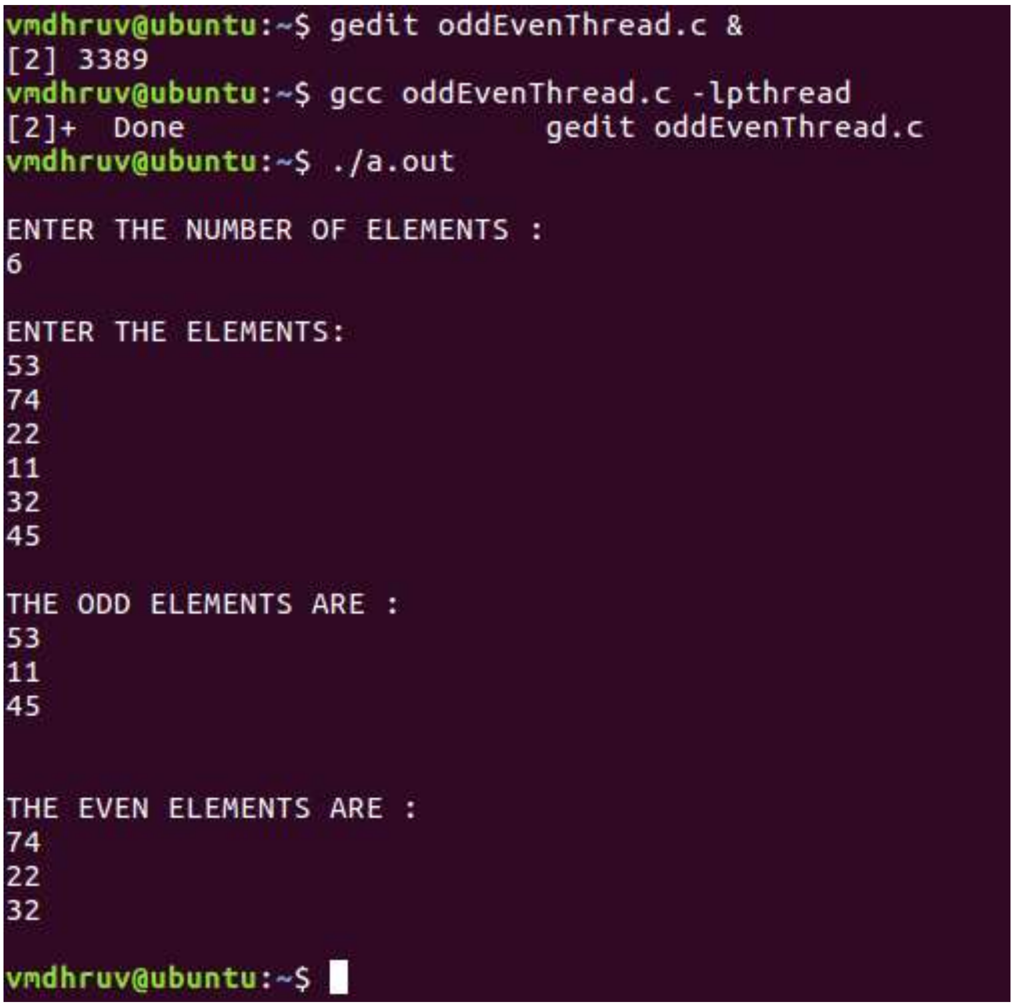
Q3. Write the above mentioned problem using threads -pthread library



oddEvenThread.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

typedef struct
{
	int A[10],n;
} array;

void *odd(void *args)
{
	array *actual_args = args;
	int i, o = 0, e = 0;
	int n = actual_args -> n;

	printf("\nTHE ODD ELEMENTS ARE : \n");
	for(i = 0; i < n; i++)
	{
		if(actual_args -> A[i] % 2 != 0)
			printf("%d\n",actual_args -> A[i]);
	}

	printf("\n");
	printf("\nTHE EVEN ELEMENTS ARE : \n");
	for(i = 0; i < n; i++)
	{
		if(actual_args -> A[i] % 2 == 0)
			printf("%d\n",actual_args -> A[i]);
	}
	printf("\n");
}



int main()
{
	int n, A[20], i;
	pthread_t odd1, even1;
	array array1;

	printf("\nENTER THE NUMBER OF ELEMENTS : \n");
	scanf("%d", &n);
	array1.n = n;

	printf("\nENTER THE ELEMENTS: \n");
	for(i=0; i < n; i++)
	{
		scanf("%d",&A[i]);
		array1.A[i] = A[i];
	}

	if(pthread_create(&odd1, NULL, odd, (void *) &array1))
		printf("error");
	pthread_join(odd1, NULL);
}
```

OUTPUT



```
vmdhruv@ubuntu:~$ gedit oddEvenThread.c &
[2] 3389
vmdhruv@ubuntu:~$ gcc oddEvenThread.c -lpthread
[2]+  Done                    gedit oddEvenThread.c
vmdhruv@ubuntu:~$ ./a.out

ENTER THE NUMBER OF ELEMENTS :
6

ENTER THE ELEMENTS:
53
74
22
11
32
45

THE ODD ELEMENTS ARE :
53
11
45


THE EVEN ELEMENTS ARE :
74
22
32

vmdhruv@ubuntu:~$ 
```

Q4. Discuss Journaling in files.

A file system in which the hard disk maintains data integrity in the event of a system crash or if the system is otherwise halted abnormally. The journaled file system (JFS) maintains a log, or journal, of what activity has taken place in the main data areas of the disk.

The basic functions/advantages of the journaled file system (JSF) are:

- **a.** Fault-resilient file system - data integrity is ensured.
- **b.** Updates to directories and bitmaps are constantly written to a serial log on disk before the original disk log is updated.
- **c.** In the event of a system failure, the JFS ensures that the data on the disk has been restored to its pre-crash configuration.
- **d.** It also recovers unsaved data and stores it in the location where it would have gone if the computer had not crashed, making it an important feature for mission-critical applications.

Journaling file systems in different operating systems - not all operating systems provide the same journaling technology.

A. **Windows NT** offers a less robust version of the full system.

   If a Windows NT system crashes, you may not lose the entire disk volume, but you will likely lose all the data that hadn't yet been written to the disk prior to the crash.

B. **Default Linux** system, ext2fs, does not journal at all.

   That means, a system crash--although infrequent in a Linux environment--can corrupt an entire disk volume.

C. **XFS** - a journaling file system from Silicon Graphics, became a part of the open-source community in 1999.

   This has had important implications for Linux developers, who previously lacked such insurance features. XFS is capable of recovering from most unexpected interruptions in less than a second, and it epitomizes the high-performance journaling file system of the future.

Q5. Consider any web application and discuss what are the possible threats and how the protection is ensured in such system through OS level.

Despite their convenience, there are drawbacks when it comes to relying on web applications for business processes. One thing all business owners have to acknowledge and guard themselves against would be the presence of software vulnerabilities and threats to web applications.

Common threats in web applications and the role of the OS is given as follows

1. <u>Security Misconfiguration</u>

Threat: A functioning web application is usually supported by some complex elements that make up its security infrastructure. This includes databases, OS, firewalls, servers and other application software or devices. All these elements require frequent maintenance and configuration to keep the web application running properly. It is advisable schedule penetration tests for the web applications to test out its capability of handling sensitive data.

Role of the OS:

a. **Firewall**

Configuring a firewall on the computer system can help tackle the threat of security misconfiguration in the web application. The operating system itself might provide a firewall, eg. Windows Firewall. One must ensure that the firewall is turned on for both – public and private networks.

b. **Post-Configuration clean up**

By turning on system protection, Windows will allow you to create system restore points so you can restore your system to a point before unwanted changes were made to your system.

2. Malware

Threat: Upon downloading malware, severe repercussions like activity monitoring, access to confidential information and backdoor access to large scale data breaches can be incurred. Malware can be categorized in different groups since they work to achieve different goals- Spyware, Viruses, Ransomware, Worms, and Trojans.

Role of the OS:

a. **Firewall**

To combat this problem, make sure to install and keep firewalls up to date. Ensure that all your operating systems have been updated as well.

b. **System back-up**

It is highly recommended that you make a back-up of your system and update it on a regular basis. Do also make sure to backup important files in external safe environments. This essentially means that if you are locked out, you will be able to access all your information without having to pay due to ransomware.

c. **Antivirus**

Good antivirus software also come built in with the operating system eg. – Windows defender. provide a great deal more protection than just searching for viruses. The antivirus must be configured to automatically scan downloaded files for identifying malicious items.

d. **Operating System patches and updates**

Perform checks on your operating system, security software, browsers used and third party plugins. If there are patches and updates for the OS/plugins, make sure to update as soon as possible.

e. **Manage App Permissions (Android OS)**

While installing an app, pay attention to the app's permissions. It will show you if the app needs access to certain parts of your mobile device. If any of the permissions look suspicious, or if you don't want to share the information, don't install the app.

3. Injection Attacks

Threat: Some examples of these attacks include SQL injection, code injection and cross site scripting. SQL injection attacks usually hijack control over the website owner's database through the act of data injection into the web application. The data injected gives the website owner's database instructions that have not been authorized by the site owner themselves. This results in data leaking, removal or manipulation of stored data. Code injection, on the other hand, involves the injecting of source codes into the web application while cross site scripting injects code (javascript) into browsers.

Role of the OS:

a. **Strong validation and user authorization settings**

To combat injection attacks, business owners are advised to implement input validation techniques and robust coding. Business owners are also encouraged to make use of 'least privilege' principles so that the user rights and authorization for actions are minimized.

4. <u>Phishing Scam</u>

Threat: Phishing scam attacks are usually involved and interfere directly with email marketing efforts. These types of threats are designed to look like emails that are from legitimate sources, with the goal of acquiring sensitive information like login credentials, bank account numbers, credit card numbers and other data. Alternatively, they can also be used to send in malware that, upon clicking, may end up gaining access to the user's information.

Role of the OS:

a. **Antivirus**

The antivirus must be configured to automatically scan downloaded files for identifying malicious items. Scanning links and information before downloading, is also provided by the operating system and antiviruses these days.

b. **Manage App Permissions (Android OS)**

While installing an app, pay attention to the app's permissions. It will show you if the app needs access to certain parts of your mobile device. If any of the permissions look suspicious, or if you don't want to share the information, don't install the app.

Besides the antivirus, the operating system here can provide only limited functionality. To prevent such incidents from happening, one must ensure that people are aware and capable of spotting suspicious emails.

5. <u>Brute Force</u>

Threat: Then there's also brute force attacks, where hackers attempt to guess passwords and forcefully gain access to the web application owner's details.

Role of the OS:

a. **Encryption**

Encryption software is very important for ensuring that physical access to hardware is difficult unless one knows the code to access it. New versions of Windows come with "Bitlocker" encryption software pre-installed. By taking the time to encrypt data, this ensures that they are difficult for hackers to make use of it for anything else unless they have encryption keys.

Android OS device will more than likely have the encryption option. This scrambles all the data on the phone - apps, media and more - until you put in the decryption password, which you will need to do every time you turn it on.

b. **Restrict the number of logins on the system**

Can deter brute force attack by limiting the number of logins one can undertake.