Date :	Exp. Title Install an operating system on a physical or logical (virtual) machine.	Page No.
Exp. No. 01		

Steps to download or Install Virtual machine:

Step 1: Check if your system is compatible. Before you can enable Hyper-V on your windows 11 machine, you need to make sure that your system is compatiable.

Step 2: Enable Hyper-V on windows 11.

Step 3: Download the windows 11 image.

Step 4: Configure the VM settings.

Step 5: Create a virtual machine by following steps.

Steps involved in installation of ubuntu

Step 1: Download the ubuntu ISO file.

Step 2: Create a bootable USB drive : you can use third-party software like Rufus to create the bootable USB drive.

Step 3: Boot from the USB flash drive.

Step 4: Install ubuntu.

Step 5: Select how you want to install ubuntu.

Date : Exp. No.	Exp. Title	Page No.

Step 6: Choose your location and set your time zone.

Step 7: Enter your name and your computer's name.

Step 8: Choose and config your password.

Note:

Before Installing ubuntu, you should back go your data to an external device. drive or cloud storage.

Installing ubuntu will over write any other os on your computer.

Author

Fork System Call:

Description:

Used to create new process. The new process consist of a copy of the original process. The value of process id form the child process is zero, where the value of process id for the parent is on integer value greather then zero.

Date:

Exp. No. 02

Exp. Title Write a C-program to implement the system call using fork().

Page No.

Syntax

fork();

Algorithum:

Step 1: Start the program.

Step 2: Declear the variables pid and child id.

Step 3: Get the child id value using system call fork.

Step 4: If child id is greather than zero then print as "I am in the parent proces".

Step 5: If child id != 0 the using pid()
system call. system get the process id.

Step 6: Print "I am in the pagent process" and print the process id.

Step 7: If child id != 0 then using .... getpid()
System call get the parent process id.

Step 8: Print "I am in the parent process" and print the parent process id.

Step 9: Else if child id value less than zero then print as "I am in the child process".

Date : Exp. No.	Exp. Title	Page No.

Step 10: If child id!=0 then using getpid() system call get the process id.

Step 11: Print "i am in the child process" and print the process id.

Step 12: If child pid =0 then using getpid() system call get the parent process id.

Step 13: Print "i am in the child process" and print the parent process id.

Step 14: Stop the program.

output:

```
i am in the parent process 4970
i am in the parent process 4970
i am in the parent process 4201
i am in child process 4970
i am in the child process 4971
```

Date :
Exp. No.

Exp. Title

Page No.

Program:

Source Code:

```
/* fork system call */

#include <stdio.h>
#include <unistd.h>
#include <sys/type.h>
int main()

{
   int id, childpid;
   id = getpid();
   if ((childpid = fork())>0)
   {
     printf("\n i am in the parent process %d\n", id);
     printf("\n i am in the child process %d ", getpid());
     printf("\n i am in the parent process %d", getppid());
   }
   else
   {
     printf("\n i am in child proces %d", id);
     printf("\n i am in the child process %d", getpid());
     printf("\n i am in the child process %d", getppid());
   }
}
```

Avitta

Wait() and Exit() System Calls:

Description:

fork():

Used to create new process. The new process consists of a copy of the address space of the original process. The value of process id for the child process is zero, where as the value of the process id for the parent is an integer value greater than zero.

Syntax: fork();

wait():

The parent waits for the child process to complete using the wait system call. The wait system call returns the process identifier of a terminated child, so that the parent can tell which of its possibly many children has terminated.

Syntax: wait (NULL);

exit():

A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the exit system call. At that point, the process may return data (output) to its parent process (via the wait system call).

Syntax: exit (0);

Date:

Exp. No. 03 (A)

Exp. Title Write a C-program to implement wait() and Exit() system call.

Page No.

Algorithm:

Step 1: Start the program.

Step 2: Declare the variable pid and i as integer.

Step 3: Get the child id value the system call fork().

Step 4: If child id where is less than zero then print "fork failed".

Step 5: Else if child id value is equal to zero, it is the id value of the child and then start the child process executed and perform steps 7 and 8.

Step 6: Else perform step 9.

Step 7: Use a for loop for almost five child process to be called.

Step 8: After execution of the loop then print "child process ends".

Step 9: Execute the system call wait() to make the parent to wait for the child process to get over.

Step 10: Once the child process are terminated the parent terminats and have prints "parent process ends".

Date : Exp. No.	Exp. Title	Page No.

Step 11: After both the parent and the child processes get terminated & execute the wait() system call to permanently get deleted from the OS.

Step 12: Stop the program.

Program:

Source Code:

```
# include <stdio.h>
# include <unistd.h>
  int main()
  {
    int i, pid;
    pid = fork();
    if (pid == -1)
    {
      printf("fork failed");
      exit(0);
    }
    else if (pid == 0)
    {
     printf("\n child process starts");
     for (i=0; i<5; i++)
     {
       printf("\n child process %d is called", i);
     }
     printf("\n child process ends");
  } }
```

Date :

Exp. No.

Exp. Title

Page

```
else
{
    wait(0);
    printf("\n Parent process ends");
}
exit(0);
}
```

Anitha

Output :

```
Child process Starts.
child process 0 is called
Child process 1 is called
child process 2 is called
Child process 3 is called
Child process 4 is called
Child process ends
parent process ends.
```

Date :

Exp. No. 03 (B)

Exp. Title Write a C-program to implement wait() and Exit() system call using switch case.

Page No.

```
# include <stdlib.h>
# include <errno.h>
# include <stdio.h>
# include <unistd.h>
# include <sys/types.h>
# include <sys/wait.h>
  int main()
  {
    pid_t pid;
    int rv;
    switch (pid = fork())
    {
      case -1:
          perror ("fork");
          exit(1);
      case 0:
          printf("\n CHILD: This is the child process:\n");
          fflush (stdout);
          printf("CHILD: My PID is %d \n", getpid());
          printf("CHILD: Enter my exit status (make small):\n");
          scanf("%d", &rv);
          printf("CHILD: I'm outta here!\n");
          exit(rv);
          default:
              printf("\n PARENT: This is the parent process\n");
              printf("PARENT: My PID is %d \n", getpid());
              fflush (stdout);
              printf("PARENT: I'm now waiting for my
                      child to exit()....\n");
          wait(&rv);
```

Output:

PARENT: This is the parent process

PARENT: My pid is 6767

PARENT: I'm now waiting for my child to exit()----

CHILD: This is the child process!

CHILD: My pid is 6758

CHILD: Enter my exit status (make it small): 1

CHILD: I'm outta here!

PARENT: My child's pid is 6758

PARENT: My child's exit status is: 1

PARENT: I'm outta here!

Date :

Exp. No.

Exp. Title

Page No.

```
        printf ("PARENT: My child's PID is %d\n", pid);
        fflush (stdout);
        printf ("PARENT: My child's exit status is: %d\n",
                WEXITSTATUS (rv));
        printf ("PARENT: I'm outta here!\n");
        exit (0);
    }
    return 0;
}
```

Author: