

| **TITLE:** System calls |
| --- |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**AIM:** To understand the working Process based system calls.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO 1.** To introduce basic concepts and functions of operating systems.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. **Silberschatz A., Galvin P., Gagne G. “Operating Systems Principles”, Willey Eight edition.**
2. **William Stallings “Operating Systems” Person, Seventh Edition**

**Edition.**

1. **Sumitabha Das “ UNIX Concepts & Applications”, McGraw Hill Second**

**Edition.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

System Calls Provide the Interface between a process and the OS.

System calls are usually made when a process in user mode requires access to a resource.

Then it requests the kernel to provide the resource via a system call.

System calls are required in the following situations −

1. If a file system requires the creation or deletion of files.
2. Reading and writing from files also require a system call.
3. Creation and management of new processes.
4. Network connections also require system calls. This includes sending and receiving packets.
5. Access to a hardware devices such as a printer, scanner etc. requires a system call.

**Description of the application to be implemented:**

**Program for System Call:**

1. Write a Program for creating process using System call (E.g fork()) Create a child process. Display the details about that process using getpid and getppid functions. In a child process, Open the file using file system calls and read the contents and display.

**Implementation details:** (printout of code / screen shot)  
  
  
import os

import sys

import errno

FILE\_PATH = "file.txt"

def handle\_error(message):

print(message, file=sys.stderr)

sys.exit(1)

def main():

process\_pid = os.getpid()

print(f"Process PID: {process\_pid}")

try:

fd = os.open(FILE\_PATH, os.O\_RDWR)

except OSError as e:

if e.errno == errno.ENOENT:

print(f"File '{FILE\_PATH}' does not exist.")

sys.exit(1)

else:

handle\_error(f"Failed to open file: {e}")

pid = os.fork()

if pid < 0:

handle\_error("Fork failed")

if pid == 0:

child\_pid = os.getpid()

parent\_pid = os.getppid()

print(f"Child Process PID: {child\_pid}")

try:

os.lseek(fd, 0, os.SEEK\_SET)

message = os.read(fd, 1024)

if message:

print(f"Child read: {message.decode()}")

else:

print("Child read: No data found.")

os.lseek(fd, 0, os.SEEK\_END)

child\_message = "Child.\n"

os.write(fd, child\_message.encode())

except OSError as e:

handle\_error(f"Error in child process: {e}")

else:

parent\_pid = os.getpid()

print(f"Parent Process PID: {parent\_pid}")

try:

os.lseek(fd, 0, os.SEEK\_END)

parent\_message = "Parent.\n"

os.write(fd, parent\_message.encode())

except OSError as e:

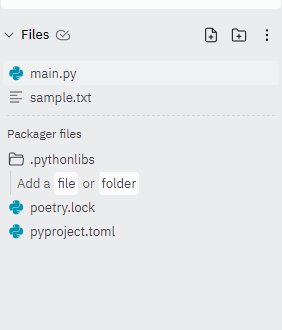
handle\_error(f"Error in parent process: {e}")

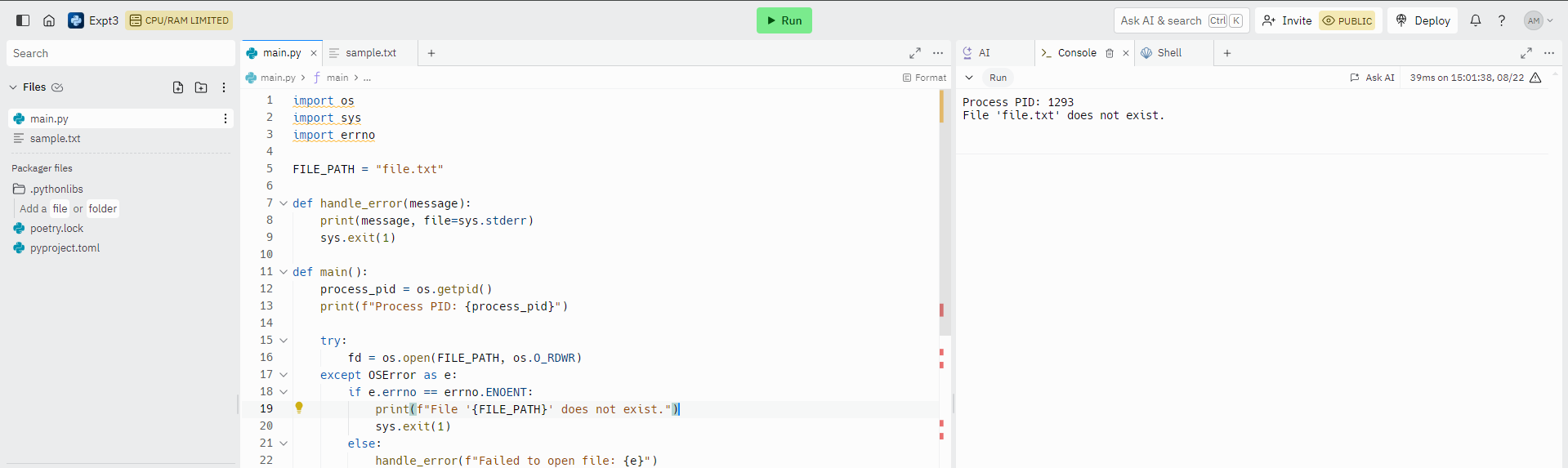
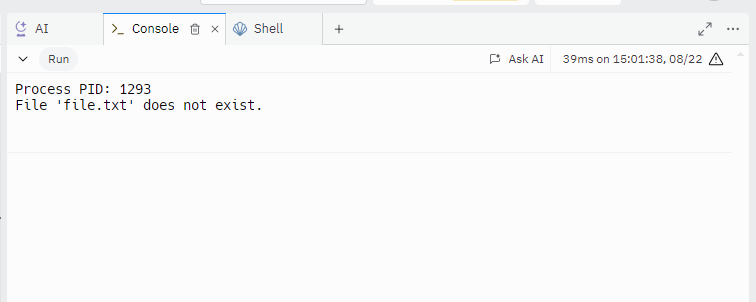
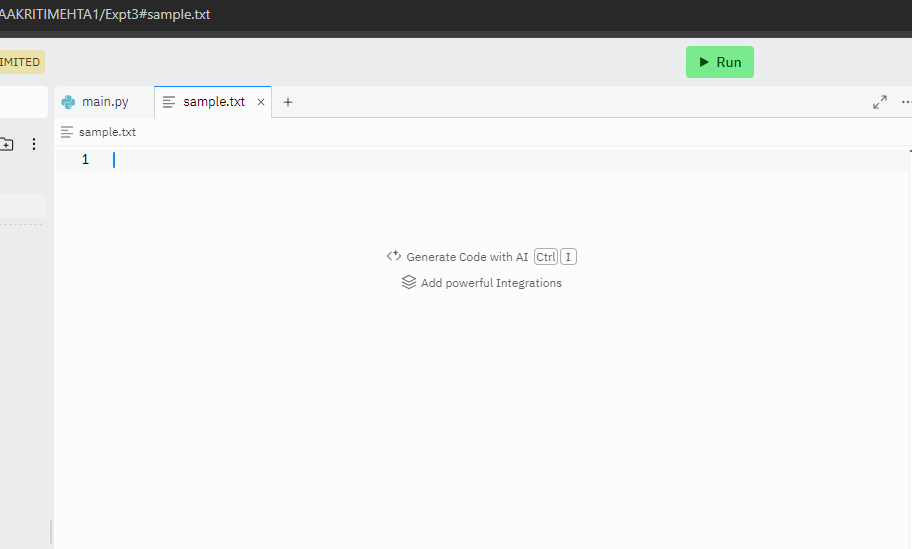
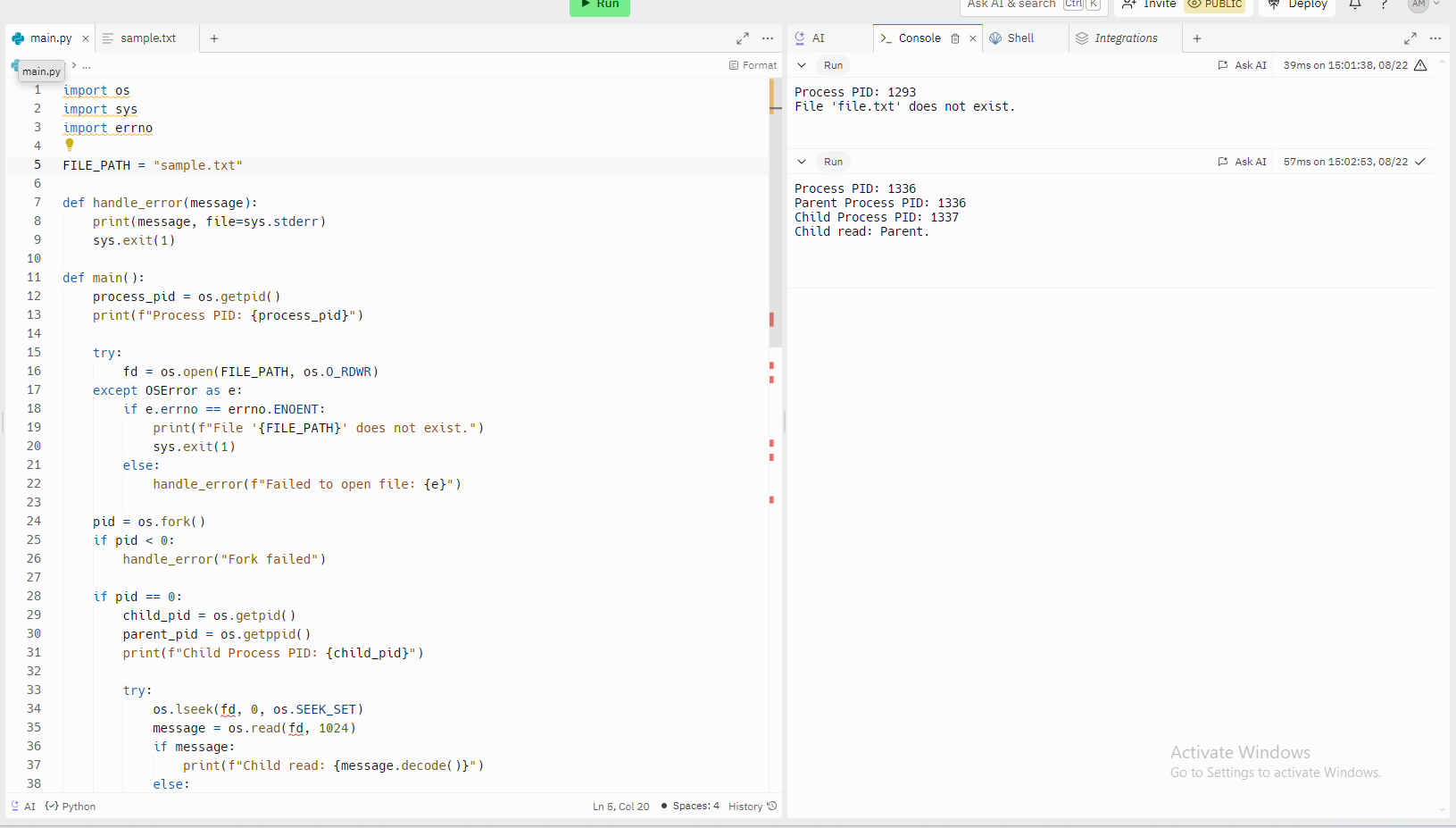
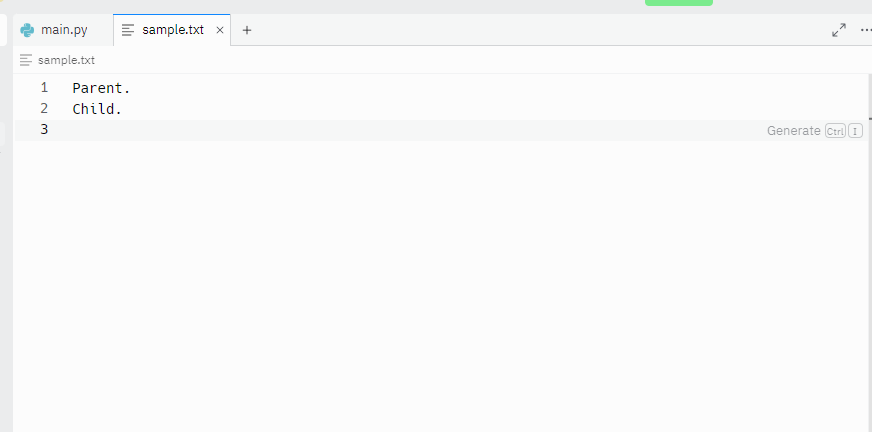
os.wait()

os.close(fd)

if \_\_name\_\_ == "\_\_main\_\_":

main()

File does not exist.  


  
  
File exists  
  
  


**Conclusion :**

Learned working process based system calls. **Post Lab Descriptive Questions**

1. Describe System Call Interface.  
   The System Call Interface is a crucial component of the operating system that provides a way for programs to interact with the kernel. It acts as a gateway between user-space applications and the underlying hardware managed by the operating system. System calls are the mechanism through which user programs request services from the operating system, such as file operations, process control, and communication.

* User Space to Kernel Space
* System Call Invocation
* Kernel Mode Execution
* Return to User Space

1. List the types of System Calls.

Process Control:

* fork(): Create a new process.
* exec(): Replace the current process image with a new process image.
* wait(): Wait for a process to change state
* exit(): Terminate the calling process.
* getpid(): Get the process ID of the current process.
* getppid(): Get the parent process ID.

File Management:

* open(): Open a file.
* read(): Read data from a file.
* write(): Write data to a file.
* close(): Close an open file descriptor.
* lseek(): Reposition the file offset.
* unlink(): Delete a file.

Device Management:

* ioctl(): Control device parameters.
* read(): Read data from a device .
* write(): Write data to a device .

Information Maintenance:

* gettimeofday(): Get the current time.
* settimeofday(): Set the system time.
* uname(): Get system information.

Communication:

* pipe(): Create a pipe for inter-process communication.
* socket(): Create a socket for network communication.
* bind(): Bind a socket to a specific address.
* listen(): Listen for incoming connections on a socket.
* accept(): Accept an incoming connection on a socket.
* send(): Send data through a socket.
* recv(): Receive data from a socket.

**Date: 22 / 8 / 24 Signature of faculty in-charge**