File Transfer Tool Using Shared Memory, Pipes, and Signals

Requirements Update:

- The **parent process** must read the file and send its content to the **child process** using **System V shared memory**.
- The **child process** must receive the file chunks from shared memory and write them to a new file.
- **Pipes** must be used for the parent process to send **file metadata** (file name, size) to the child process.
- **Signals** must be used by the child process to notify the parent process of **progress updates** and **completion**.

Detailed Process Flow:

1. **Parent Process:**
    - Opens the file to be transferred.
    - Sends **file metadata** (name, size) to the child process using a **pipe**.
    - Writes **1KB chunks of file data** to **System V shared memory**.
    - Waits for **progress signals** from the child process.
2. **Child Process:**
    - Reads the **file metadata** from the pipe.
    - Receives **1KB chunks of data** from shared memory.
    - Writes the received chunks to a new file.
    - Sends **SIGUSR1 signals** to the parent process for each chunk processed and a **SIGUSR2 signal** when the transfer is complete.

Hints:

- Use `fork()` to create a child process.
- Use `pipe()` to establish a unidirectional pipe for metadata transfer.
- Use **System V shared memory** (`shmget()`, `shmat()`, `shmdt()`) for file data transfer.
- Use `kill()` to send signals from the child process to the parent process.

Real-Time System Monitoring Tool Using Shared Memory, Pipes, and Signals

---

Requirements Update:

- The **parent process** must handle **real-time logging** of system stats.
- The **child process** must continuously monitor **CPU and memory usage** from the `/proc` filesystem and write the data to **System V shared memory**.
- **Pipes** must be used to send **metadata** (timestamps, process ID, etc.) from the child process to the parent process.
- **Signals** must be used by the child process to notify the parent process when new data is available.

Detailed Process Flow:

1. **Parent Process:**
    - Initializes **System V shared memory** for storing system stats.
    - Waits for **SIGUSR1 signals** from the child process indicating new data is available.
    - Reads the stats from shared memory and writes them to **`system_log.txt`**.
    - Handles **SIGINT** to terminate both processes gracefully.
2. **Child Process:**
    - Continuously reads **CPU and memory usage** from **`/proc/stat`** and **`/proc/meminfo`**.
    - Writes the stats to **System V shared memory**.
    - Sends **metadata** (timestamp, process ID) to the parent process using a **pipe**.
    - Sends **SIGUSR1 signals** to the parent process whenever new data is written to shared memory.

Hints:

- Use **`pipe()`** to send metadata from the child process to the parent process.
- Use **signal handlers** to handle **SIGUSR1** and **SIGINT** signals.