北京交通大学操作系统实验报告

姓名：程维森

学号：21231264

一. 实验内容

这个项目的目标是练习各种进程间通信方法（用于数据传递和同步），并学习Map-Reduce（并行计算）。这两者是工业界经常使用的非常重要的技术。

详细内容：

这个项目由三个独立的子项目组成，每个子项目执行相同的任务：给定一个文本文件，程序输出包含特定单词的行。例如，对于一行"Hello World!"，假设感兴趣的单词是"world"，那么这行应该被输出。（请注意，如果一行是"Hello worlds"，则该行不应该被输出）。你的程序（即父进程）将创建一个子进程。父进程可以打开文本文件，读取内容，并使用以下

其中一种方法将其传递给子进程，但不能检查单词。而子进程不能打开该文件但可以检查单词。最终，父进程（而不是子进程）应该按字母顺序输出这些行。。

要求:

1. 使用管道作为传递文件内容和结果的方法。
2. 使用 Unix 域套接字作为传递文件内容和结果的方法。
3. 使用共享内存作为传递文件内容和结果的方法。此外，子进程创建 4 个线程，每个线程充当一个 Mapper；子进程的主线程充当单一的 Reducer。Map-Reduce 仅适用于这个子项目。不允许使用 Hadoop 作为 Map-Reduce 基础设施,相反，您必须使用 Posix 线程编程来实现 Map-Reduce

二. Use Pipe as the method of passing the file content and the result

程序分析：程序的目标是是在一个文本文件中查找特定单词，如果找到，就将包含该单词的行和行号输出到一个文件中。它使用了管道进行进程间通信，并且包括了一个父进程和一个子进程。

1. 父进程：
   - 创建了一个管道，用于和子进程进行通信。
   - 打开了一个文本文件（big.txt），逐行读取文件内容。
   - 对于每一行，它检查是否包含特定的单词（在这里是"call"）。如果找到，父进程将该行的内容写入管道，供子进程读取。
   - 如果文件中没有找到特定单词，父进程向管道写入消息"Word not found"。
   - 等待子进程结束。
2. 子进程：
   - 从管道中读取数据，这些数据是父进程写入的包含特定单词的行。
   - 将读取到的数据写入一个输出文件(output.txt)中，同时加上"Child Process Output: "前缀。
   - 当读取结束后，关闭文件和管道，完成任务。

父进程分析:

```
else { // 父进程
  close(pipefd[0]); // 父进程关闭读取端

  ifstream inputFile("big.txt"); // 打开文件
  if (!inputFile) {
      cerr << "File not found!" << endl;
      close(pipefd[1]);
      return 1;
  }

  string wordToFind = "call"; // 需要在文件中查找的单词
  string line;
  int lineNumber = 0;
  bool found = false;

  // 逐行读取文件内容，查找特定单词
  while (getline(inputFile, line)) {
      lineNumber++;
      if (line.find(wordToFind) != string::npos) {
          found = true;
          // 输出单词和行数
          cout << "Word found: " << wordToFind << " at line number: " << lineNumber << endl;
          // 将结果写入管道，供子进程读取
          write(pipefd[1], line.c_str(), line.size());
      }
  }

  // 如果没有找到特定单词，向子进程发送消息
  if (!found) {
      write(pipefd[1], "Word not found", sizeof("Word not found"));
  }

  // 关闭写入端和文件
  close(pipefd[1]);
  inputFile.close();
```

父进程：

创建管道：在父进程开始执行时，它创建了一个管道（pipefd[2]），用于和子进程进行通信。

打开文件和逐行读取： 父进程打开了一个文本文件（big.txt）并逐行读取文件内容。查找特定单词： 对于每一行，父进程检查是否包含特定的单词（"call"）。如果找到，它将

该行的内容写入管道，供子进程读取。如果文件中没有找到特定单词，父进程向管道写入消息"Word not found"。

等待子进程结束： 父进程在完成文件读取后，关闭了管道的写入端，并且等待子进程结束。

子进程：

```
if (pid == 0) { // 子进程
    close(pipefd[1]); // 子进程关闭写入端

    ofstream outputFile("output.txt"); // 打开输出文件

    char buffer[100];
    ssize_t bytesRead;

    // 循环读取管道数据，直到读取结束
    while ((bytesRead = read(pipefd[0], buffer, sizeof(buffer))) > 0) {
        outputFile << "Child Process Output: " << buffer << endl;
    }

    // 关闭文件和管道
    outputFile.close();
    close(pipefd[0]);

    cout << "Output written to output.txt." << endl;
```

子进程：

关闭写入端：子进程在创建后，关闭了管道的写入端，表示它只从管道中读取数据。

从管道读取数据：子进程从管道中读取数据，这些数据是父进程写入的包含特定单词的行或者"Word not found"消息。

将数据写入文件：子进程将读取到的数据写入一个输出文件(output.txt)中，每行前面加上"Child Process Output: "前缀。

关闭资源：当读取结束后，子进程关闭了输出文件和管道的读取端。

实验结果展示：

仅仅展示部分结果，全部的实验结果请参考附件的’output.txt’文件

实验总结:

这个管道（pipe）实验是一个简单的进程间通信示例，用于演示父子进程之间的数据传递。以下是这个实验的一些总结要点:

1. 目标： 这个实验的目标是演示如何使用管道进行进程间通信。它创建了一个父进程和一个子进程，父进程从一个文本文件中查找特定单词，并将匹配的行写入管道，而子进程从管道中读取这些数据，并将它们写入输出文件。

2. 父子进程：父进程和子进程是两个独立的进程，它们之间可以通过管道传递数据。父进程负责文件操作和数据的写入，而子进程负责读取管道中的数据并将其写入输出文件。

3. 管道创建：通过调用 `pipe` 函数，创建了一个匿名管道，用于在父子进程之间传递数据。

4. 进程间通信： 父进程逐行读取文本文件内容，查找指定单词，并将匹配的行写入管道。子进程从管道中读取数据，将其写入输出文件。

5. 父子进程协作： 父进程在完成文件读取后关闭管道的写入端，子进程在创建后关闭了管道的写入端，以确保不再有数据写入管道。

6. 资源管理： 在父子进程结束后，需要关闭文件和管道，以释放资源。

7. 问题： 这个示例存在一些问题，包括无法处理多个匹配行的情况，以及进程间同步问题。实际应用中，可能需要更复杂的逻辑来处理这些情况。

总的来说, 这个实验有助于理解管道的基本概念和如何使用它在不同进程之间传递数据。然而，实际应用中，通常需要更复杂的逻辑来解决更具挑战性的问题。

三. Use Unix Domain Socket as the method of passing the file content and the result.

这段代码的设计功能是建立一个使用 Unix 域套接字的简单客户端-服务器通信系统。这个系统用于在一个文本文件中查找特定的单词，并将找到的内容写入另一个文件。以下是代码的主要功能步骤：

1. 服务器端（父进程）：
   - 创建一个 Unix 域套接字, 并将其绑定到文件系统中的路径 (通过`SOCKET_PATH`定义）。
   - 监听来自客户端的连接请求。
   - 创建一个子进程来处理客户端的请求。
   - 在父进程中，服务器套接字等待并接受客户端的连接。
   - 打开一个名为`big.txt`的文件，读取文件内容，并在文件中搜索名为`call`的单词。
   - 如果在文件的某一行中找到了该单词, 则将这行内容通过套接字发送给客户端 (子进程）。
   - 如果在文件中未找到该单词，则发送一个消息`"Word not found"`给客户端。
   - 关闭服务器套接字并等待子进程结束。
2. 客户端（子进程）：
   - 创建一个 Unix 域客户端套接字，并连接到服务器套接字。
   - 打开（或创建）一个名为`output.txt`的文件，用于存储从服务器接收的数据。
   - 从服务器读取数据（服务器发送的文件中包含单词`call`的行），并将其写入`output.txt`文件中。
   - 一旦读取完成，关闭客户端套接字和输出文件，并打印消息说明输出已写入`output.txt`。
3. 清理:
   - 在所有操作完成后，父进程将删除由服务器创建的`socket.sock`文件。

这个程序可以用于在大型文本文件中查找某个单词并将结果输出到另一个文件中，演示了 Unix 域套接字的 IPC 机制，并展示了如何在父子进程之间进行通信。此外，它也展示了进程创建（fork）、文件操作和基本的错误处理。在实际使用中，为了确保程序的稳健性和安全性，还需要更多的错误检查和异常处理逻辑。

```
} else { // 父进程
    // 等待客户端连接
    int clientSocket = accept(serverSocket, NULL, NULL);
    if (clientSocket == -1) {
        cerr << "Accept failed!" << endl;
        close(serverSocket);
        return 1;
    }

    ifstream inputFile("big.txt"); // 打开文件
    if (!inputFile) {
        cerr << "File not found!" << endl;
        close(clientSocket);
        close(serverSocket);
        return 1;
    }

    string wordToFind = "call"; // 需要在文件中查找的单词
    string line;
    int lineNumber = 0;
    bool found = false;

    // 逐行读取文件内容，查找特定单词
    while (getline(inputFile, line)) {
        lineNumber++;
        if (line.find(wordToFind) != string::npos) {
            found = true;
            // 输出单词和行数到套接字，供子进程读取
            write(clientSocket, line.c_str(), line.size());
        }
    }

    // 如果没有找到特定单词，向子进程发送消息
    if (!found) {
        write(clientSocket, "Word not found", sizeof("Word not found"));
    }

    // 关闭套接字和文件
    close(clientSocket);
    close(serverSocket);
    inputFile.close();

    wait(NULL); // 等待子进程结束
}
```

客户端父程序

```
    // 创建Unix域套接字
    clientSocket = socket(AF_UNIX, SOCK_STREAM, 0);
    if (clientSocket == -1) {
        cerr << "Socket creation failed!" << endl;
        return 1;
    }

    // 设置服务器地址
    memset(&serverAddress, 0, sizeof(struct sockaddr_un));
    serverAddress.sun_family = AF_UNIX;
    strncpy(serverAddress.sun_path, "./unix", sizeof(serverAddress.sun_path) - 1);

    // 连接到服务器
    if (connect(clientSocket, (struct sockaddr *)&serverAddress, sizeof(struct sockaddr_un)) == -1)
        cerr << "Connection failed!" << endl;
        close(clientSocket);
        return 1;
    }

    int lineNumber;
    ssize_t bytesRead = recv(clientSocket, &lineNumber, sizeof(int), 0);

    // 处理从服务器接收到的数据
    if (bytesRead > 0) {
        if (lineNumber == -1) {
            cout << "Word not found in the file." << endl;
        } else {
            cout << "Word found in line number: " << lineNumber << endl;
        }
    } else {
        cerr << "Error receiving data from server!" << endl;
    }

    // 关闭套接字
    close(clientSocket);

    return 0;
}
```

服务端子程序

实验总结: 这个实验展示了一个使用 Unix 域套接字进行进程间通信的简单例子。

它模拟了一个客户端-服务器场景，其中服务器进程在本地文本文件中查找特定的单词，并将包含该单词的文本行发送给客户端进程。客户端进程接收这些数据并将它们写入另一个文件。以下是实验的关键步骤和特点的总结:

1. Unix 域套接字的创建和使用:

- 服务器端创建了一个 Unix 域套接字并与一个路径绑定。
- 客户端也创建了一个 Unix 域套接字并连接到服务器。

2. 进程创建和通信:

   - 程序使用`fork()`创建了一个子进程，子进程作为客户端，父进程作为服务器。

   - 服务器和客户端通过 Unix 域套接字进行通信。

3. 文件操作:

   - 服务器读取本地文件`big.txt`，搜索包含特定单词`call`的行。

   - 客户端将接收到的数据写入`output.txt`文件。

4. 信号和等待子进程:

   - 父进程使用`wait(NULL)`等待子进程完成，保证客户端进程先运行完成。

5. 错误处理:

   - 程序在关键操作（如创建套接字、绑定、监听、连接、接收和发送数据）失败时进行错误检查，并输出错误信息。

6. 资源管理：

   - 在操作完成后，套接字被关闭，相关文件被删除。

通过这个实验，可以学习到如何在 Linux 环境下使用系统调用来进行进程间通信，以及如何管理进程、文件和网络资源。它也是理解 IPC 机制、客户端-服务器模型和并发编程概念的一个实用示例。此外，这个例子还凸显了在进行系统编程时，资源清理和错误处理的重要性。

四．Use Shared Memory as the method of passing the file content and the result.

程序分析：这段代码实现了一个使用共享内存进行进程间通信的程序，主要逻辑如下：

1. 共享内存创建：程序首先创建了一个共享内存段，大小为 4MB，使用`shmget`函数。如果创建失败，程序输出错误信息并退出。

2. 创建子进程：接着，程序使用`fork`函数创建了一个子进程。如果创建失败，程序同样输出错误信息并退出。

3. 子进程逻辑：

- 子进程通过`shmat`函数将共享内存附加到自身的内存空间，得到一个指向共享内存的指针。
- 子进程尝试打开文件"big.txt"，如果文件不存在，输出错误信息并退出。然后，它读取文件内容到共享内存，并在末尾加上 null 终止符，以确保字符串结束。
- 子进程查找共享内存中包含特定单词（"call"）的所有行，将这些行的内容和行号输出到文件"output.txt"中。
- 子进程完成后，分离共享内存（`shmdt`函数）并删除共享内存段（`shmctl`函数）。

4. 父进程逻辑：

- 父进程等待子进程执行完毕（`wait`函数）。

总的来说，该程序的目标是在一个大文件中查找指定的单词（"call"），使用共享内存实现了父子进程间的数据传递。父进程负责等待子进程的完成。这个程序

假设共享内存足够大，能够容纳整个文件的内容。

```
if (pid == 0) { // 子进程
    char* sharedMemory = static_cast<char*>(shmat(shmid, NULL, 0));
    if (sharedMemory == (char*)-1) {
        cerr << "Shared memory attachment failed!" << endl;
        return 1;
    }

    ifstream inputFile("big.txt"); // 打开文件
    if (!inputFile) {
        cerr << "File not found!" << endl;
        shmdt(sharedMemory); // 分离共享内存
        shmctl(shmid, IPC_RMID, NULL); // 删除共享内存段
        return 1;
    }

    // 读取文件内容到共享内存
    inputFile.read(sharedMemory, SHM_SIZE);
    inputFile.close();
    sharedMemory[SHM_SIZE - 1] = '\0'; // 确保共享内存以 null 结尾

    string wordToFind = "call"; // 需要在文件中查找的单词
    string content(sharedMemory);

    // 找到所有包含指定单词的行，并将这些行的内容和行号输出到output.txt文件中
    ofstream outputFile("output.txt");
    size_t pos = content.find(wordToFind, 0);
    size_t lineNumber = 1;
    while (pos != string::npos) {
        // 找到单词所在的行的开始位置
        size_t lineStart = content.rfind('\n', pos) + 1;
        // 找到单词所在的行的结束位置
        size_t lineEnd = content.find('\n', pos);
        if (lineEnd == string::npos) {
            lineEnd = content.size();
        }
        // 提取单词所在的行的内容
        string lineContent = content.substr(lineStart, lineEnd - lineStart);
        // 输出单词所在的行的内容和行号
        outputFile << "Line " << lineNumber << ": " << lineContent << endl;

        // 继续查找下一个单词
        pos = content.find(wordToFind, pos + 1);
        lineNumber++;
    }

    outputFile.close();
    shmdt(sharedMemory); // 分离共享内存
    shmctl(shmid, IPC_RMID, NULL); // 删除共享内存段
```

子进程代码

实验结果展示：

具体请参考实验附录:

output.txt

```
Line 1: "There will call upon you to-night, at a quarter to eight o'clock," it said, "a gentleman who desires to consult you upon a matter of the very deepest moment. Your recent services to one of the royal houses of Europe have shown that you are one who may safely be trusted with matters which are of an importance which can hardly be exaggerated. This account of you we have from all quarters received. Be in your chamber then at that hour, and do not take it amiss if your visitor wear a mask."
Line 2: "You had my note?" he asked with a deep harsh voice and a strongly marked German accent. "I told you that I would call." He looked from one to the other of us, as if uncertain which to address.
Line 3: "You will excuse this mask," continued our strange visitor. "The august person who employs me wishes his agent to be unknown to you, and I may confess at once that the title by which I have just called myself is not exactly my own."
Line 4: "Then, good-night, your Majesty, and I trust that we shall soon have some good news for you. And good-night, Watson," he added, as the wheels of the royal brougham rolled down the street. "If you will be good enough to call to-morrow afternoon at three o'clock I should like to chat this little matter over with you."
Line 5: "Oh, she has turned all the men's heads down in that part. She is the daintiest thing under a bonnet on this planet. So say the Serpentine-mews, to a man. She lives quietly, sings at concerts, drives out at five every day, and returns at seven sharp for dinner. Seldom goes out at other times, except when she sings. Has only one male visitor, but a good deal of him. He is dark, handsome, and dashing, never calls less than once a day, and often twice. He is a Mr. Godfrey Norton, of the Inner Temple. See the advantages of a cabman as a confidant. They had driven him home a dozen times from Serpentine-mews, and knew all about him. When I had listened to all they had to tell, I began to walk up and down near Briony Lodge once more, and to think over my plan of campaign.
Line 6: As he spoke the gleam of the sidelights of a carriage came round the curve of the avenue. It was a smart little landau which rattled up to the door of Briony Lodge. As it pulled up, one of the loafing men at the corner dashed forward to open the door in the hope of earning a copper, but was elbowed away by another loafer, who had rushed up with the same intention. A fierce quarrel broke out, which was increased by the two guardsmen, who took sides with one of the loungers, and by the scissors-grinder, who was equally hot upon the other side. A blow was struck, and in an instant the lady, who had stepped from her carriage, was the centre of a little knot of flushed and struggling men, who struck savagely at each other with their fists and sticks. Holmes dashed into the crowd to protect the lady; but, just as he reached her, he gave a cry and dropped to the ground, with the blood running freely down his face. At his fall the guardsmen took to their heels in one direction and the loungers in the other, while a number of better dressed people, who had watched the scuffle without taking part in it, crowded in to help the lady and to attend to the injured man. Irene Adler, as I will still call her, had hurried up the steps; but she stood at the top with her superb figure outlined against the lights of the hall, looking back into the street.
Line 7: "Our quest is practically finished. I shall call with the King to-morrow, and with you, if you care to come with us. We will be shown into the sitting-room to wait for the lady, but it is probable that when she comes she may find neither us nor the photograph. It might be a satisfaction to his Majesty to regain it with his own hands."
Line 8: "Our quest is practically finished. I shall call with the King to-morrow, and with you, if you care to come with us. We will be shown into the sitting-room to wait for the lady, but it is probable that when she comes she may find neither us nor the photograph. It might be a satisfaction to his Majesty to regain it with his own hands."
Line 9: "And when will you call?"
Line 10: "Indeed! My mistress told me that you were likely to call. She left this morning with her husband by the 5:15 train from Charing Cross for the Continent."
Line 11: "We shall see." He pushed past the servant and rushed into the drawing-room, followed by the King and myself. The furniture was scattered about in every direction, with dismantled shelves and open drawers, as if the lady had hurriedly ransacked them before her flight. Holmes rushed at the bell-pull, tore back a small sliding shutter, and, plunging in his hand, pulled out a photograph and a letter. The photograph was of Irene Adler herself in evening dress, the letter was superscribed to "Sherlock Holmes, Esq. To be left till called for." My friend tore it open, and we all three read it together. It was dated at midnight of the preceding night and ran in this way:
Line 12: "MY DEAR MR. SHERLOCK HOLMES,--You really did it very well. You took me in completely. Until after the alarm of fire, I had not a suspicion. But then, when I found how I had betrayed myself, I began to think. I had been warned against you months ago. I had been told that, if the King employed an agent, it would certainly be you. And your address had been given me. Yet, with all this, you made me reveal what you wanted to know. Even after I became suspicious, I found it hard to think evil of such a dear, kind old clergyman. But, you know, I have been trained as an actress myself. Male costume is nothing new to me. I often take advantage of the freedom which it gives. I sent John, the coachman, to watch you, ran upstairs, got into my walking clothes, as I call them, and came down just as you departed.
Line 13: "We both thought the best resource was flight, when pursued by so formidable an antagonist; so you will find the nest empty when you call to-morrow. As to the photograph, your client may rest in peace. I love and am loved by a better man than he. The King may do what he will without hindrance from one whom he has cruelly wronged. I keep it only to safeguard myself, and to preserve a weapon which will always secure me from any steps which he might take in the future. I leave a photograph which he might care to possess; and I remain, dear Mr. Sherlock Holmes,
Line 14: I had called upon my friend, Mr. Sherlock Holmes, one day in the autumn of last year and found him in deep conversation with a very stout, florid-faced, elderly gentleman with fiery red hair. With an apology for my intrusion, I was about to withdraw when Holmes pulled me abruptly into the room and closed the door behind me.
Line 15: "You did, Doctor, but none the less you must come round to my view, for otherwise I shall keep on piling fact upon fact on you until your reason breaks down under them and acknowledges me to be right. Now, Mr. Jabez Wilson here has been good enough to call upon me this morning, and to begin a narrative which promises to be one of the most singular which I have listened to for some time. You have heard me remark that the strangest and most unique things are very often connected not with the larger but with the smaller crimes, and occasionally, indeed, where there is room for doubt whether any positive crime has been committed. As far as I have heard, it is impossible for me to say whether the present case is an instance of crime or not, but the course of events is certainly among the most singular that I have ever listened to. Perhaps, Mr. Wilson, you would have the great kindness to recommence your narrative. I ask you not merely because my friend Dr. Watson has not heard the opening part but also because the peculiar nature of the story makes me anxious to have every possible detail from your lips. As a rule, when I have heard some slight indication of the course of events, I am able to guide myself by the thousands of other similar cases which occur to my memory. In the present instance I am forced to admit that the facts are, to the best of my belief, unique."
Line 16: " 'And he is admirably suited for it,' the other answered. 'He has every requirement. I cannot recall when I have seen anything so fine.' He took a step backward, cocked his head on one side, and gazed at my hair until I felt quite bashful. Then suddenly he plunged forward, wrung my hand, and congratulated me warmly on my success.
Line 17: " 'What do you call purely nominal?'
Line 18: "I was staggered, sir. I did not know what to do. Then I called at the offices round, but none of them seemed to know anything about it. Finally, I went to the landlord, who is an accountant living on the ground floor, and I asked him if he could tell me what had become of the Red-headed League. He said that he had never heard of any such body. Then I asked him who Mr. Duncan Ross was. He answered that the name was new to him.
Line 19: "We shall endeavour to clear up these points for you. And, first, one or two questions, Mr. Wilson. This assistant of yours who first called your attention to the advertisement--how long had he been with you?"
Line 20: "So far I had got when we went to visit the scene of action. I surprised you by beating upon the pavement with my stick. I was ascertaining whether the cellar stretched out in front or behind. It was not in front. Then I rang the bell, and, as I hoped, the assistant answered it. We have had some skirmishes, but we had never set eyes upon each other before. I hardly looked at his face. His knees were what I wished to see. You must yourself have remarked how worn, wrinkled, and stained they were. They spoke of those hours of burrowing. The only remaining point was what they were burrowing for. I walked round the corner, saw the City and Suburban Bank abutted on our friend's premises, and felt that I had solved my problem. When you drove home after the concert I called upon Scotland Yard and upon the chairman of the bank directors, with the result that you have seen."
Line 21: "Yes, my stepfather. I call him father, though it sounds funny, too, for he is only five years and two months older than myself."
Line 22: "I see. Then at the gasfitters' ball you met, as I understand, a gentleman called Mr. Hosmer Angel."
Line 23: "Yes, sir. I met him that night, and he called next day to ask if we had got home all safe, and after that we met him--that is to say, Mr. Holmes, I met him twice for walks, but after that father came back again, and Mr. Hosmer Angel could not come to the house any more."
Line 24: "To the Leadenhall Street Post Office, to be left till called for. He said that if they were sent to the office he would be chaffed by all the other clerks about having letters from a lady, so I offered to typewrite them, like he did his, but he wouldn't have that, for he said that when I wrote them they seemed to come from me, but when they were typewritten he always felt that the machine had come between us. That
```

实验总结:

优点:

共享内存提供了高效的进程间通信方式，适用于大量数据的传输。

使用共享内存可以避免数据的复制，提高了程序执行效率。

缺点:

共享内存需要进行适当的同步，以免多个进程同时修改共享数据导致不一致性。

对共享内存的操作需要谨慎，否则可能引发内存访问错误。

经验教训:

在使用共享内存时，确保正确处理同步问题，避免出现竞态条件。

在操作共享内存时，确保不会越界访问或者访问已经被分离的共享内存。

五. 运行时间对比与分析

管道时间：68s

共享内存：2s

差异原因：

管道（Pipes）：

管道提供了一个半双工的通信方式，通常用于有父子关系的进程间通信。

管道使用内核缓冲区来传递消息，数据必须先写入内核，然后另一个进程从内核读取，这导致上下文切换。

管道的性能通常不如共享内存和 Unix 域套接字，因为它涉及多次数据复制和上下文切换。

Unix 域套接字（Unix Domain Sockets）：

Unix 域套接字提供全双工通信，适用于任何两个进程间的通信。

性能比管道更高，因为 Unix 域套接字可以使用较少的系统调用和上下文切换。

与管道一样，Unix 域套接字仍然涉及将数据从用户空间复制到内核空间，然后再复制到另一个用户空间。

共享内存（Shared Memory）：

共享内存是最快的 IPC 机制之一，因为它允许多个进程直接访问同一内存区域。

它几乎不需要系统调用或上下文切换，从而最小化了延迟。

共享内存通信通常需要同步机制，如信号量，来协调对共享资源的访问。

六. 关于 Map-Reduce

MapReduce 是一个编程模型和一个用于处理和生成大数据集的相关实现。它由 Google 提出，并被用于很多分布式计算系统，最著名的实现是 Apache Hadoop。

MapReduce 模型主要包括两个步骤：Map（映射）步骤和 Reduce（归约）步骤。

1. Map（映射）阶段：

- 这个阶段将输入数据集分成小的片段，这些片段分别由多个映射任务并行处理。
- 每个映射任务处理一小块数据，并产生中间键值对（key-value pairs）作为输出。
- 例如，如果要计算大量文档中单词的出现频率，Map 函数会遍历每个文档，输出每个单词及其出现的次数（通常是 1）。

2. Reduce（归约）阶段：

- 在这个阶段，中间键值对的列表被合并处理，这些列表来自 Map 阶段的输出。
- 每个 Reduce 任务处理一组共享同一个键的值，然后将这些值合并成更小的值集合，通常是通过某种形式的汇总，如求和或列表合并。
- 在上述例子中，Reduce 函数会对所有映射任务输出的同一个单词的计数值进行汇总，以得到该单词的总频率。

MapReduce 模型的关键优势在于它的可扩展性和容错性。它可以在成千上万的处理器上运行，并且能够处理节点故障。当一个节点失败时，系统会自动重新安排其上的工作到其他节点上。这使得 MapReduce 非常适合于大规模数据处理任务。

由于其简单性和效率, MapReduce 已经成为大数据和分布式计算领域的重要工具。然而，对于需要频繁数据共享的任务，MapReduce 可能不是最优选择，因为每个阶段的输出都需要被写入到磁盘中，这可能会导致效率低下。因此，对于某些类型的计算任务，其他模型，如 Spark 的 RDD（弹性分布式数据集），可能更加适合。

Hadoop 是一个由 Apache 基金会开发的开源框架，它允许使用简单的编程模型对大型数据集进行分布式处理。Hadoop 是基于 Google 的 MapReduce 算法构建的，并使用 Hadoop 分布式文件系统（HDFS）来存储数据。Hadoop 的核心是设计用来支持从单个服务器到成千上万台机器的扩展性，每台机器都提供局部计算和存储。

七. 关于 Hadoop

Hadoop 框架主要包含以下几个模块：

1. Hadoop Common：这是框架的基础部分，提供了 Hadoop 的核心库和必要的 Java 文件和脚本，以便运行 Hadoop。

2. Hadoop Distributed File System (HDFS)：一个高度容错的系统，设计用于在廉价的（或者标准的）硬件上运行，并提供高吞吐量来访问应用程序数据。

3. Hadoop YARN：一个资源管理平台，负责管理计算资源在集群中的分配，并使用它们来调度用户的应用程序。

4. Hadoop MapReduce：一个 YARN 上运行的系统，负责并行处理大数据集。

Hadoop 的核心优势在于它的高可靠性和可扩展性。它能够检测并处理应用层面的故障，保证集群中的数据和应用程序能持续运行。其文件系统将数据分散存储在整个集群中，而 MapReduce 框架则在需要的时候，将计算任务分发到数据所在的节点上，这样减少了网络传输的需求，并提高了处理速度。

除了这些核心组件，Hadoop 生态系统还包括了许多其他工具，比如：

- Hive：一个建立在 Hadoop 上的数据仓库，提供了 SQL-like 的查询语言（HQL），让用户能够执行查询并将 SQL 转换成 MapReduce、Tez 或 Spark 作业。

- Pig：一个高级平台，用于创建 MapReduce 程序使用的 Pig Latin 语言。

- HBase：一个非关系型分布式数据库（NoSQL），运行在 HDFS 之上，为大型稀疏数据集提供实时读/写访问。

- Spark：一个使用了内存计算技术，比 MapReduce 更快的大数据处理工具，它也可以运行在 YARN 上。

由于其能力强大的处理大规模数据集的能力，Hadoop 已经成为大数据分析的一个重要工具，尤其是在数据挖掘、机器学习和预测分析等领域。

八. Linux 提供的不同进程间通信方法与何时使用哪种方法

Linux 操作系统提供了多种进程间通信（IPC）机制，以便在不同的进程之间交换数据。不同的 IPC 机制适用于不同的场景和需求。以下是一些常见的 IPC 方法以及它们的使用情况：

1. 管道（Pipes）和命名管道（Named Pipes）：
   - 使用场景：当需要在有父子关系的进程之间进行简单的通信时，管道是一个好选择。如果进程没有父子关系，或者你想在不相关的进程之间进行通信，可以使用命名管道（也称为 FIFOs）。

2. 信号（Signals）：
   - 使用场景：信号是一种简单的通信方式，用于通知进程发生了某个事件（如中断信号）。信号通常用于处理异常情况或者控制进程（如终止进程）。

3. 消息队列（Message Queues）：
   - 使用场景：当你需要在多个进程之间传递数据块时，消息队列是有用的。它们通过消息队列标识符进行通信，并允许非实时的、复杂的通信。

4. 共享内存（Shared Memory）：
   - 使用场景：当你需要在进程之间快速共享大量数据时。共享内存是最快的 IPC 方式，因为数据不需要在进程间复制，但是需要额外的同步机制（如信号量）。

5. 信号量（Semaphores）
   - 使用场景：主要用于同步，如控制多个进程对共享资源的访问。它们可以防止资源冲突和竞态条件。

6. 套接字（Sockets）：
   - 使用场景：套接字不仅可以用于网络通信，也可以用于在同一台机器上运行的进程之间的通信（使用 Unix 域套接字）。它们特别适合于需要流式通信的复杂通信场景。

7. 文件:
   - 使用场景：所有进程都可以访问文件系统，因此文件可以用作进程间通信的中介。但是，由于涉及到磁盘 IO，文件不适合高速通信。

8. 套接字对（Socket pairs）：
   - 使用场景：仅限于两个进程间的双向通信。这是一种特殊类型的套接字，适用于需要快速通信的父子进程。

在选择进程间通信机制时，你需要考虑以下因素：

- 性能需求：如果性能是关键因素，共享内存通常是最佳选择。

- 数据复杂性：如果你需要传递复杂的数据结构，消息队列或套接字可能是更好的选择。

- 同步需求：如果需要同步，信号量或文件锁可以帮助管理对资源的访问。
- 耦合程度：如果进程高度独立，命名管道或套接字可能更合适；对于紧密相关的进程，管道可能就足够了。
- 可靠性：信号量和消息队列提供了一定程度的可靠性，可以保证消息的送达和顺序。

每种 IPC 机制都有其特点和最佳使用场景，选择合适的机制取决于应用程序的具体需求。

九. 代码与仓库

请参考：https://github.com/sksx085/OSHomework