به نام خدا
داکیومنت توضیح پروژه فاز اول درس برنامه نویسی چند هسته ای - دکتر سوادی
موضوع پروژه: جستجوی متن در فایل های بزرگ
اعضای گروه: امیررضا سالاری منش، حمیدرضا حسینی

مقدمه

در این پروژه، مسئله جستجوی رشته‌ای در فایل‌های متنی بزرگ را بررسی کردیم. هدف، مقایسه‌ی دو پیاده‌سازی بود:

- نسخه‌ی سریال (تک نخی)
- نسخه‌ی مولتی‌ترد (چندنخی)

از زبان C++ برای بخش‌های پردازشی و از Python برای تولید داده و اسکریپت اجرای تست‌ها استفاده شده است.

مرحله ۱: تولید داده‌های آزمایشی با Python

برای شبیه‌سازی جستجوی واقعی، با استفاده از `create_data.py` فایل‌هایی بزرگ تولید کردیم.

ساخت فایل‌های متنی شامل میلیون‌ها خط داده تصادفی با رشته‌هایی خاص (مثل OpenAI).

```
def generate_text_file(file_name, line_count, keyword, frequency):
    with open(file_name, 'w') as f:
        for i in range(line_count):
            if i % frequency == 0:
                line = f"some random data... {keyword} ...more data\n"
            else:
                line = ''.join(random.choices(string.ascii_letters + " ",
k=80)) + '\n'
 f.write(line)
```

اجرای فایل:

```
python create_data.py
```

مرحله ۲: پیاده‌سازی الگوریتم سریال با C++

فایل `single_thread.cpp` نسخه ساده و تک‌ریسمانی الگوریتم را پیاده‌سازی می‌کند.

```
std::ifstream file(file_name);
std::string line;
while (std::getline(file, line)) {
    if (line.find(keyword) != std::string::npos) {
        ++count;
    }
}
```

زمان‌سنجی:

```
auto start = std::chrono::high_resolution_clock::now();
// اجرای الگوریتم
auto end = std::chrono::high_resolution_clock::now();
std::chrono::duration<double> duration = end - start;
```

مرحله ۳: پیاده‌سازی نسخه مولتی‌ترد با C++

در `multi_thread.cpp` از چند ترد برای پردازش بخش‌های مختلف فایل استفاده می‌شود.

طراحی کلی:

1. خواندن فایل به صورت کامل.

2. تقسیم فایل به n بخش.
3. اجرای جستجو در هر بخش توسط تردها.
4. ترکیب نتایج با استفاده از `std::mutex`.

```
void search_in_lines(std::vector<std::string>& lines, const std::string&
keyword, int& local_count) {
    for (const auto& line : lines) {
        if (line.find(keyword) != std::string::npos) {
            ++local_count;
        }
    }
}
```

اجرای تردها:

```
std::vector<std::thread> threads;
for (int i = 0; i < num_threads; ++i) {
    threads.emplace_back(search_in_lines, std::ref(line_blocks[i]),
std::ref(keyword), std::ref(counts[i]));
}
```

هماهنگی نتایج:

```
for (int i = 0; i < num_threads; ++i) {
    threads[i].join();
    total += counts[i];
}
```

مرحله ۴: اجرای خودکار و تحلیل نتایج با `main.py`

اسکریپت `main.py` اجرای هر دو نسخه و مقایسه‌ی زمانی آن‌ها را بر عهده دارد.

```
start = time.perf_counter()
os.system(f"./single_thread {file_name} {search_term}")
duration = time.perf_counter() - start
print(f"Serial took {duration:.2f} seconds")
```

بررسی نتیجه:

اما در اخر میایم و نتیجه این کار را با دو روش بررسی میکنیم و نتیجه رو میتونیم مقایسه کنیم.

روش سریال:

```
Found in ./generated_files/file_28.txt at line 33100
Found in ./generated_files/file_28.txt at line 42931
Found in ./generated_files/file_28.txt at line 44520
Found in ./generated_files/file_28.txt at line 44988
Found in ./generated_files/file_28.txt at line 47242
Total occurrences of "streahte": 2278
Time taken: 22.9752 seconds
```

که همونطور که مشاهده میکنید جستجوی کلمه streahte در بین فایل ها ۲۲ ثانیه طول کشیده.

تصویر زیر اما جستجوی همان کلمه در همان فایل ها اما با تعداد ۵ ترد است که همانطور که مشخص است زمان به طور قابل توجهی کاهش یافته است و به ۶ ثانیه رسیده است.

```
[Thread 2] Found in ./generated_files/file_194.txt at line 48790
[Thread 1] Found in ./generated_files/file_28.txt at line 44520
[Thread 1] Found in ./generated_files/file_28.txt at line 44988
[Thread 1] Found in ./generated_files/file_28.txt at line 47242

Total occurrences of "streahte": 2278
Time taken with OpenMP (5 threads): 6.81224 seconds
```