کد ارائه شده یک برنامه کاربردی Multi-threadبه زبان c است که هدف آن انجام عملیات مختلف و بررسی انواع فایل در یک دایرکتوری مشخص می‌باشد.

این برنامه تعداد فایل های با پسوندهای خاص مانند .txt، .png و jpg. فایل های با پسوندهای دیگر را می شمارد و نمایش می دهد. همچنین تعداد کل فایل های دایرکتوری را تعیین و اندازه پوشه ریشه را محاسبه و علاوه بر این، بزرگترین و کوچکترین فایل ها را در دایرکتوری پیدا می کند.

برنامه با درخواست ورودی از کاربر یک directory path می گیرد و عملیات زیر را مانیتور می کند:

1. تعداد کل فایل‌ها
2. تعداد انواع فایل‌ها (چند نوع فایل داریم)
3. آدرس مربوط به فایل با بیشترین سایز + مقدار سایز آن
4. آدرس مربوط به فایل با کمترین سایز + مقدار سایز آن
5. سایز نهایی فولدر ریشه

ورودی نمونه:

Directory path: /Users/negar/Desktop/German

خروجی نمونه:

Total .txt files: 0

Total .png files: 3

Total .jpg files: 1

Total other files: 13

Total files: 429616224

Largest file size: 188225628 bytes

Largest file path: /Users/negar/Desktop/German/Video-20231017\_165940-Meeting Recording.mp4

Smallest file size: 6148 bytes

Smallest file path: /Users/negar/Desktop/German/.DS\_Store

**بررسی کد برنامه:**

قفل pthread\_mutex\_t، این متغیر یک قفل mutex است که توسط کتابخانه «pthread» ارائه شده است. برای اطمینان از ایمنی رشته هنگام به روز رسانی متغیرهای تعداد فایل استفاده می شود. قفل mutex برای جلوگیری از دسترسی و تغییر همزمان چندین رشته به متغیرها استفاده می شود که می تواند منجر به خراب شدن داده ها شود.

ساختار "ThreadArgs" حاوی یک آرایه کاراکتر "directoryPath" با اندازه 100 است. این ساختار برای ارسال آرگومان ها به تابع رشته "calculateFileTypes()" استفاده می شود که اجازه می دهد تا هر رشته یک مسیر دایرکتوری منحصر به فرد برای پردازش دریافت کند.

* **تابع calculateRootFolderSize**

تابع در ابتدا پارامتر «rootFolderPath» را می گیرد که نشان دهنده مسیر پوشه ریشه است که باید محاسبه شود.

متغیری به نام "totalSize" داریم که این متغیر اندازه تجمعی همه فایل ها و زیر پوشه ها را در پوشه ریشه ذخیره می کند.

برای هر ورودی، تابع با 'rootFolderPath' و نام ورودی، مسیر کامل را می سازد.

تابع 'stat' برای بازیابی اطلاعات مربوط به فایل یا زیرپوشه مشخص شده توسط مسیر ساخته شده استفاده می شود. اگر تابع 'stat' از کار بیفتد، یک پیام خطا چاپ می شود و حلقه به ورودی بعدی ادامه می یابد.

اگر ورودی یک فایل معمولی باشد (یعنی یک دایرکتوری یا فایل خاص نباشد)، اندازه آن به متغیر "totalSize" اضافه می شود و اگر ورودی یک دایرکتوری باشد، یک فراخوان بازگشتی به تابع «calculateRootFolderSize» انجام می‌شود و مسیر ورودی را به‌عنوان «rootFolderPath» جدید عبور می‌دهد.

پس از پردازش همه ورودی‌های پوشه ریشه، تابع «totalSize» محاسبه‌شده را چاپ می‌کند.

* **تابع findLargestAndSmallestFileSize**

این تابع پارامتر "directoryPath" را می گیرد که نشان دهنده مسیر دایرکتوری مورد جستجو است.

دو آرایه کاراکتری، "largestFilePath" و "smallestFilePath" برای ذخیره مسیرهای بزرگترین و کوچکترین فایل ها را به ترتیب داریم.

تابع 'stat' برای بازیابی اطلاعات مربوط به فایل مشخص شده توسط مسیر فایل ساخته شده استفاده می شود. اگر تابع 'stat' از کار بیفتد، یک پیام خطا چاپ می شود و حلقه به ورودی بعدی ادامه می یابد.

اگر فایل یک فایل معمولی باشد (یعنی یک دایرکتوری یا فایل خاص نیست)، اندازه آن با بزرگترین و کوچکترین اندازه های فعلی مقایسه می شود. اگر اندازه فایل بزرگتر از بزرگترین اندازه فعلی باشد، متغیرهای "largestSize" و "largestFilePath" به روز می شوند. اگر اندازه فایل کوچکتر از کوچکترین اندازه فعلی باشد، متغیرهای «smallestSize» و «smallestFilePath» به روز می شوند.

پس از پردازش تمام ورودی های داکیومنتری، تابع بزرگترین و کوچکترین اندازه فایل را به همراه مسیرهای فایل مربوطه چاپ می کند.

* **تابع sharedLogic**

این تابع حاوی منطق مشترک برای شمارش فایل‌ها با پسوندهای خاص و به‌روزرسانی متغیرهای شمارش فایل است.

اگر ورودی یک فایل معمولی باشد (نه دایرکتوری)، پسوند فایل را بررسی می کند. اگر فایل دارای پسوند باشد، پسوند را با پسوندهای از پیش تعریف شده (.txt, .png, .jpg, .jpeg) مقایسه می کند. بر اساس پسوند، متغیر تعداد فایل مربوطه ('totaltxtCount'، 'totalpngCount'، 'totaljpgCount'، یا 'totalotherCount') را با به دست آوردن و آزاد کردن قفل mutex ('pthread\_mutex\_lock()' و `pthread\_mutex\_lock) افزایش می دهد.

اگر ورودی یک دایرکتوری باشد به صورت بازگشتی تابع «sharedLogic()» را با مسیر فهرست جدید («filePath») فراخوانی می کند تا محتویات آن را پردازش کند.

* **calculateFileTypes**

هدف آن فراخوانی تابع 'sharedLogic()' برای محاسبه تعداد انواع فایل های مختلف در دایرکتوری ما هست.

تابع «sharedLogic()» با «directoryPath» که از ساختار «ThreadArgs» به دست می‌آید فراخوانی می‌شود. این تابع ساختار دایرکتوری را طی می کند، متغیرهای تعداد فایل ها ('totaltxtCount', 'totalpngCount', 'totaljpgCount', 'totalotherCount') را به روز می کند.

* **main() function**

از کاربر خواسته می شود مسیر دایرکتوری را وارد کند و ورودی خوانده می شود که ورودی را در متغیر "directoryPath" ذخیره می کنیم.

اگر ورودی یک زیر شاخه باشد، یک thread جدید با استفاده از «pthread\_create» ایجاد می شود. تابع «calculateFileTypes» به‌عنوان روال شروع برای رشته ارسال می‌شود و «directoryPath» زیر شاخه به عنوان آرگومان با استفاده از آرایه «threadArgs» ارسال می‌شود.

هنگامی که همه رشته‌ها ایجاد شدند، رشته اصلی منتظر می‌ماند تا هر رشته با استفاده از «pthread\_join» به پایان برسد و اطمینان حاصل کند که برنامه پیش از موعد خارج نمی‌شود و پس از اتمام تمام رشته ها، تابع «closedir» برای بستن دایرکتوری اصلی فراخوانی می شود.

در نهایت تابع "pthread\_mutex\_destroy" برای پاک کردن قفل mutex فراخوانی می شود.