**شرح پروژه سیستم عامل**

این کد به زبان C نوشته شده که به منظور جمع‌آوری و آمار اطلاعات فایل‌ها و دایرکتوری‌ها از یک ساختار فایل سیستم استفاده می‌کند. در ادامه، توضیحاتی در مورد اجزاء اصلی و عملکرد برنامه آورده شده:

**ساختارها و متغیرها:**

struct FileInfo: یک ساختار که نام و اندازه فایل را نگهداری می‌کند.

pthread\_mutex\_t mutex : یک mutex برای همگام‌سازی دسترسی به منابع اشتراکی.

struct FileInfo \*allFiles: یک اشاره‌گر به آرایه‌ای از اطلاعات فایل‌ها.

int \*fileCount: یک اشاره‌گر به تعداد کل فایل‌ها.

char \*fileTypes[MAX\_FILE\_TYPES]: یک آرایه از اشاره‌گرها به رشته‌های نوع فایل.

int fileTypeCounts[MAX\_FILE\_TYPES]: تعداد فایل‌های هر نوع.

struct MessageType: یک ساختار برای ارسال پیام‌ها درون صف مرسلی.

**توابع اصلی:**

getFileType(char \*path): تابعی که نوع فایل را از نام فایل استخراج می‌کند.

addFileType(char \*fileType): تابعی برای اضافه کردن یک نوع فایل به آرایه و برگرداندن اندیس آن.

traverseDirectory(char \*path, int msgqid, int shmid, int processIndex, pthread\_mutex\_t \*mutex): تابع بازگرداننده اطلاعات فایل‌ها و دایرکتوری‌ها در یک دایرکتوری.

processFileOrDir(char \*path, int msgqid, int shmid, int processIndex, pthread\_mutex\_t \*mutex): تابع پردازش فایل یا دایرکتوری.

calculateFileCountAndSize(struct FileInfo \*files, int numFiles): تابع محاسبه آمار کلی فایل‌ها و اندازه‌های آن‌ها.

main(): تابع اصلی که مسیر دایرکتوری اصلی را از کاربر دریافت می‌کند و سپس فرآیندها را شروع می‌کند و پس از اتمام همه، نتایج را نمایش می‌دهد.

**استفاده از Shared Memory و Message Queue:**

از Shared Memory برای اشتراک اطلاعات بین فرآیندها(آرایه allFiles و fileCount) استفاده شده است.

از Message Queue برای ارسال تعداد فایل‌های هر نوع از هر فرآیند فرعی به فرآیند اصلی استفاده شده است.

همگام‌سازی:

Mutex به عنوان یک وسیله همگام‌سازی برای اطمینان از ایمنی دسترسی به منابع اشتراکی مورد استفاده قرار گرفته است.

برنامه از یک الگوریتم پویا برای مدیریت اندازه آرایه فایل‌ها و همچنین نوع فایل‌ها استفاده کرده است. همچنین از اصول fork برای ایجاد فرآیندهای جدید برای هر زیردایرکتوری استفاده می‌کند.