پروژه میان‌ترم

شبیه‌ساز فایل سیستم UNIX

رسیدیم به پروژه‌ی جذاب میان‌ترم، جایی که تمام مهارت‌های کدنویسی شما و دانشی که از ابتدای ترم یاد گرفته‌اید به چالش کشیده می‌شود. در این پروژه، شما باید بخشی کوچک از فایل سیستم سیستم‌عامل‌های مبتنی بر یونیکس (Unix-based) را شبیه‌سازی و پیاده‌سازی کنید.

فایل سیستم چیست؟

فایل سیستم روشی است که **کامپیوتر** برای ذخیره و سازماندهی اطلاعات روی حافظه‌هایی مثل هارد دیسک یا فلش استفاده می‌کند. وقتی یک فایل را روی کامپیوتر ذخیره می‌کنید، فایل سیستم مشخص می‌کند که این فایل در کجای حافظه قرار بگیرد و چگونه بازیابی شود. همچنین امکان دسته‌بندی اطلاعات در پوشه‌ها، تنظیم دسترسی کاربران و مدیریت فضای ذخیره‌سازی را فراهم می‌کند. بدون فایل سیستم، داده‌ها به‌صورت نامرتب روی حافظه قرار می‌گرفتند و دسترسی به آن‌ها غیرممکن می‌شد. نگران نباشید؛ اینها تنها توضیحات مربوط به فایل سیستم است و شما قرار نیست تمام چیزهایی که گفته شد را پیاده‌سازی کنید.
فایل سیستم علاوه بر دادهٔ اصلی، متادیتا (مانند اندازه، تاریخ ایجاد، مجوزهای دسترسی و ...) را نیز نگه می‌دارد تا سیستم‌عامل بتواند عملیات مدیریتی را انجام دهد.

اهمیت فایل سیستم چیست؟

فایل سیستم یکی از بخش‌های اساسی هر سیستم عامل است، زیرا بدون آن، ذخیره‌سازی و مدیریت داده‌ها به شکل کارآمد امکان‌پذیر نخواهد بود. این سیستم امکان دسترسی سریع و سازمان‌یافته به اطلاعات، جلوگیری از تداخل داده‌ها، مدیریت فضای ذخیره‌سازی، و حفظ امنیت و یکپارچگی فایل‌ها را فراهم می‌کند. همچنین، با استفاده از فایل سیستم، کاربران می‌توانند داده‌های خود را در پوشه‌ها مرتب کنند، فایل‌ها را جستجو کنند و از قابلیت‌هایی مانند مجوزهای دسترسی و بازیابی اطلاعات استفاده کنند. در نهایت، بدون یک فایل سیستم مناسب، عملکرد کلی کامپیوتر مختل شده و کار با داده‌ها بسیار دشوار خواهد شد. بسیاری از ویژگی‌های پیشرفتهٔ سیستم‌عامل‌ها (مثل snapshot و journaling) وابسته به توانایی‌های فایل سیستم هستند.

فایل سیستم در لینوکس

در لینوکس، فایل سیستم به‌صورت یک ساختار درختی سازمان‌دهی شده که از ریشه (/) شروع می‌شود. ریشه بالاترین سطح در فایل سیستم است و تمام پوشه‌ها و فایل‌ها زیرمجموعه‌ی آن قرار می‌گیرند. در لینوکس دو نوع آدرس‌دهی برای دسترسی به فایل‌ها وجود دارد:

- **آدرس‌دهی مطلق (Absolute Path):** از ریشه (/) شروع می‌شود، مثل `/home/user/document.txt` که نشان می‌دهد فایل `document.txt` در پوشه‌ی `user` داخل `home/` قرار دارد.
- **آدرس‌دهی نسبی (Relative Path):** از همان پوشه‌ای که در آن هستید شروع می‌شود، مثلاً اگر در `/home/user` باشید، برای دسترسی به `document.txt` کافی است بنویسید `document.txt` یا `./document.txt` .

این ساختار باعث می‌شود دسترسی به فایل‌ها و مدیریت آن‌ها در لینوکس ساده و کارآمد باشد.

الزامات پروژه

مقدمه

در این پروژه لازم است اصول **برنامه‌نویسی شیءگرا (OOP)** را به‌صورت منسجم و مکمل یکدیگر به کار بگیرید تا محصول نهایی، ساختاری مرتب، مقیاس‌پذیر و توسعه‌پذیر داشته باشد. پایبندی به قواعد **کدنویسی تمیز (Clean Code)** نیز حیاتی است؛ به‌بیان دیگر، کد باید روان، مستند و خالی از پیچیدگی‌های اضافی باشد تا در آینده بتوان آن را بدون دردسر نگه‌داری یا گسترش داد. هدف، پیاده‌سازی یک شبیه‌ساز کوچک از فایل‌سیستم Unix است؛ ابزاری خطّ فرمان که با چند دستور ساده امکان ساخت، مشاهده، ویرایش، جابه‌جایی، کپی و حذف پوشه‌ها و فایل‌های متنی با پسوند `.txt` را برای کاربر فراهم می‌کند.

مدیریت مسیر (Path Management)

در این پروژه امکان استفاده از مسیرهای مطلق و نسبی برای دسترسی به فایل‌ها و فولدرها باید فراهم شود. این ویژگی به کاربر اجازه می‌دهد که بدون محدودیت در سلسله‌مراتب پوشه‌ها، به راحتی به فایل‌های مورد نظر

خود دسترسی داشته باشد. همچنین بهتر است تابعی برای جستجوی فایل یا فولدر بر اساس مسیر داده‌شده پیاده‌سازی شود که پردازش مسیرها را تسهیل می‌کند.

در صورت واردکردن مسیر نامعتبر، پیام خطای مناسبی مثل `Path not found` نمایش دهید.

ایجاد و حذف فولدرها (Directories)

کاربران باید بتوانند با استفاده از دستور

```
mkdir <path> <folder_name>
```

فولدرهای جدیدی را در مسیرهای مشخص‌شده ایجاد کنند. اگر مسیر مشخص نشود، فولدر در دایرکتوری فعلی ساخته می‌شود. همچنین دستور

```
rm <path>
```

برای حذف فولدرها در نظر گرفته شده است که امکان مدیریت بهتر فضای ذخیره‌سازی را فراهم می‌کند.

ایجاد و حذف فایل‌ها (Files)

کاربران باید امکان ایجاد فایل‌های متنی جدید را داشته باشند که این کار با دستور

```
touch <path> <file_name>.txt
```

انجام خواهد شد. اگر مسیر مشخص نشود، فایل در دایرکتوری جاری ساخته می‌شود. برای حذف فایل‌ها نیز باید دستور

```
rm <path>
```

پیاده‌سازی شود تا کاربر بتواند فایل‌های غیرضروری را مدیریت کند.

پیمایش بین فولدرها (Navigation)

باید دستوری برای تغییر مسیر بین فولدرها پیاده‌سازی شود. دستور<path> cd امکان جابه‌جایی بین مسیرهای مختلف را فراهم می‌کند. همچنین باید قابلیت بازگشت به فولدر قبلی با دستور:

```
cd ..
```

وجود داشته باشد. برای مشاهده فایل‌ها و فولدرهای داخل دایرکتوری فعلی نیز باید دستور ls پیاده‌سازی شود تا کاربر بتواند محتویات پوشه‌ها را بررسی کند.

مدیریت محتوا و ویرایش فایل‌ها

کاربران باید بتوانند فایل‌های متنی خود را ویرایش کنند. برای این کار، دستوری مانند:

```
nwfiletxt <path>
```

این دستور محتوای فایل را عوض میکند و آن را جدید مینویسد بعد از زدن این دستور باید ورودی هارا در خط های متوالی بگیرید. نحوه خروج از گرفتن متن، به دست خودتان است.

برای ایجاد یک فایل و افزودن متن به آن در نظر گرفته می‌شود. همچنین، باید امکانی برای اضافه کردن متن جدید به انتهای فایل‌های موجود با استفاده از دستور:

```
appendtxt <path>
```

وجود داشته باشد. برای تغییر محتوای یک خط خاص از فایل، دستور:

```
editline <path> <line> <text>
```

باید پیاده‌سازی شود و در صورتی که نیاز به حذف یک خط خاص باشد، دستور:

```
deline <path> <line>
```

این کار را انجام دهد. علاوه بر این، برای مشاهده محتوای فایل‌ها باید دستور:

```
cat <path>
```

در نظر گرفته شود.

مدیریت انتقال و کپی فایل/فولدر

کاربران باید قادر باشند فایل‌ها و فولدرهای خود را بین مسیرهای مختلف جابه‌جا کنند. این قابلیت باید با دستور:

```
mv <source_path> <destination_path>
```

پیاده‌سازی شود تا امکان انتقال فایل‌ها و فولدرها فراهم گردد. همچنین، دستور:

```
cp <source_path> <destination_path>
```

باید برای کپی کردن فایل‌ها و فولدرها اجرا شود تا کاربران بتوانند از داده‌های خود نسخه‌ای دیگر ایجاد کنند.

توجه کنید که وقتی از دستوری cp استفاده میکنید نباید همان آبجکت کپی شده را در فولدر مقصد بگذارید بلکه باید کپی شده اش را در مقصد بگذارید به این معنا که اگر تغییری در یکی از آن ها داده شد در دیگری تغییری داده نشود.

تغییر نام فایل و فولدر

کاربران باید بتوانند فایل‌ها و فولدرهای خود را تغییر نام دهند به طوری که هیچ تغییری توی محتویات فایل یا فولدر نباشد. این کار باید با استفاده از دستور

```
rename <path> <new_name>
```

انجام شود. این قابلیت کمک می‌کند تا کاربران فایل‌های خود را بهتر سازماندهی کرده و از نام‌های مناسب‌تری برای مدیریت داده‌های خود استفاده کنند.

مثال دستورات در محیط اجرای برنامه

```
$ mkdir parsa
$ ls
parsa folder
$ cd parsa
parsa$ ls
parsa$ touch test.txt
File 'test.txt' created in the current directory.
parsa$ nwfiletxt test.txt
enter the lines (/end/ means done)
this
is
a
test
/end/
parsa$ cat test.txt
this
is
a
test
parsa$ appendtxt notes.txt
file was not found
parsa$ appendtxt test.txt
enter the lines (/end/ means done)
add
this
to test
/end/
parsa$ ls
test.txt
parsa$ cat test.txt
this
is
a
test
add
this
to test
```

```
parsa$ cd ..
$ ls
parsa folder
```

برای اطلاعات بیشتر و نحوهٔ عملکرد دستورات، بهتر است خودتان نیز تحقیق کنید؛ علاوه بر آن می‌توانید از سایت https://www.terminaltemple.com استفاده کنید.

نتیجه‌گیری

این پروژه یک شبیه‌ساز ساده از مدیریت فایل در Unix را ارائه می‌دهد که کاربران می‌توانند فایل‌ها و فولدرها را ایجاد، ویرایش، حذف و جابجا کنند. این سیستم به درک بهتر ساختارهای سیستم فایل Unix کمک می‌کند و می‌تواند علاوه بر قوی کردن مهارت‌های برنامه‌نویسی، دید عملی شما را نسبت به مباحث سیستم‌عامل گسترش دهد.

پس از ایجاد ریپازیتوری پروژه، آن را عمومی کنید و لینک آن را ارسال کنید.