**DOKUMENTACJA**

**W projekcie brali udzial:**

Przemek Jopa

Konrad Szymański

**Uruchomienie projektu**

W celu uruchomienia programu należy przejść do katalogu, w którym się znajduje. Następnie z pozycji terminalu należy wybrać komendę

make all

aby skompilować całość.

Demon może zostać uruchomiony manualnie za pomocą polecenia

./exc <nazwa\_katalogu\_źródłowego> <nazwa\_katalogu\_docelowego>

jeśli przebywamy w katalogu programu. Po dodaniu do polecenia opcji -R demon będzie rekurencyjnie synchronizował katalogi, czyli sprawdzał wszystkie podkatalogi.

W zależności od rozmiaru plików wykonywane są oddzielne metody kopiowania. Próg dzielący pliki małe od dużych może być przekazywany jako opcjonalny argument w postaci komendy

./exc <nazwa\_katalogu\_źródłowego> <nazwa\_katalogu\_docelowego> <prog\_dzielacy>

Istnieje możliwość natychmiastowego obudzenia się demona poprzez wysłanie mu sygnału SIGUSR1 używając komendy

killall -USR1 exc

w nowym oknie terminalu.

**Testowanie**

W celu przetestowania programu należy przejść do katalogu, w którym się znajduje i uruchomić skrypt testujący wprowadzając w terminalu polecenie

./skrypt

Podczas trwania skryptu tworzone są dwa katalogi „katalog11” (źródłowy) oraz „katalog22” (docelowy). W katalogu źródłowym dodawane są dwa pliki tekstowe „plik1.txt” i „plik2.txt” z przykładową zawartością oraz tworzone są dwa podkatalogi „katalog11\_A” i „katalog11\_B”. W pierwszym podkatalogu dodawany jest plik tekstowy „plik3.txt” z przykładową zawartością. Po tych działaniach kopiowany jest duży plik „DuzyPlik.bin” do „katalog11”.

Następuje uruchomienie demona synchronizującego katalog źródłowy i docelowy. Wysyłany jest sygnał SIGUSR1, po którym uruchamiany jest drugi demon rekurencyjnie synchronizujący katalogi. Po tych czynnościach działanie skryptu zostaje zakończone.

W programie zostało zaimplementowanych kilka funkcji do zarządzania plikami, katalogami oraz funkcjami systemowymi. Zaczerpnięte zostały z kilku bibliotek języka „C”. Poniżej jest spis najważniejszych z nich:

|  |  |
| --- | --- |
| **Ctype.h** | |
| Isdigit() | jest to funkcja sprawdzająca czy podany znak jest cyfrą dziesiętną |
| **Sys/stat.h** | |
| Mkdir() | Jest to funkcja tworząca nowy katalog |
| Stat() | funkcja zwraca informacje o pliku o podanej ścieżce |
| Lstat() | funkcja identyczna do stat(). Różnica polega na tym, że jeśli ścieżka wskazuje na plik będący dowiązaniem symbolicznym to wyświetli informacje o pliku będącym dowiązaniem. Natomiast stat() przekaże informacje o pliku docelowym na który wskazuje dowiązanie. |
| **Errno.h** - *jest to mechanizm zgłaszania błędów. Jeśli program zwróci „-1” lub „null”, czyli kod błędu, program zagląda do globalnej zmiennej errno w celu odnalezienia liczby, której jest przyporządkowany konkretny błąd* | |
| Dirent.h | |
| Closedir() | funckja zamykająca strumień katalogowy |
| Opendir() | funkcja otwierająca strumień dla wskazanego katalogu |
| Readdir() | zwaraca wskaźnik na plik w katalogu. Pliki znajdują się w kolejce i każde wywołanie tej funkcji powoduje przeszukiwanie wszystkich plików w danym katalogu. |
| **Syslog.h** – *zawiera w sobie definicje błędów systemowych, które są zapisywane w pliku* | |
| Signal.h | |
| Sigaction() | pozwala na zmianę działania procesu, gdy system otrzyma określony sygnał |
| Sigemptyset() | funkcja zawieszająca działanie sygnałów, które zostały przekazane do funkcji |
| **sys/mmn.h** – *biblioteka do zarządzania deklaracjami pamięci* | |
| Mmap() | funkcja odwzorowująca fragment pliku w przestrzeni adresowej, dzięki czemu można odwoływać siędo niego jak do tablicy. Plik jest mapowany w wielokrotnościach rozmiaru strony. Więc plik jest wczytywany do pamięci stronami. Dzięki użyciu tej funkcji skracamy czas operacji na dużych plikach, gdyż nie wykorzystujemy funkcji systemowych, które są dosyć wolne. |
| **fcntl.h** - *bliblioteka do manipulowania deskryptorem plików, pozwala na dostęp do flag plików oraz mechanizmu blokad POSIX* | |
| Open() | funkcja przekształcająca ścieżkę pliku na deskryptor pliku |
| **unistd.h** – *udostępnia funkcje niezbędne do tworzenia programów, które wykorzystują usługi systemu operacyjnego* | |
| Exit() | odpowiada za wyłączenie programu oraz usunięcie pamięci, którą dana aplikacja wykorzystywała |
| Unlink() | usuwa odniesienie do ścieżki będącej dowiązaniem symbolicznym |
| Pause() | wstrzymuje działanie procesu do momentu otrzymania sygnału wznawiającego |
| Write() | zapisywanie danych do pliku |
| Alarm() | opóźnienie wykonywania dalszych instrukcji na konkretny okres czasu podawany w sekundach. Może zostać przerwane otrzymanym sygnałem z zewnątrz w dowolnym momencie |
| Close() | funkcja zamykająca deskryptor pliku otwarty przy pomocy funkcji open(). |

**Lista zaimplementowanych funkcji**

Program rozpoczyna swoje działanie poprzez zainicjowanie zmiennych oraz sprawdzenie ilości wprowadzonych parametrów. Następnie program jest obsługiwany przez cyklicznie wykonujące się funkcje zarządzające całym procesem synchronizacji dwóch katalogów.

1. **Int WhatIsIt(struct stat info)**

*Ta funkcja jest wykorzystywana do sprawdzenia, czy podana ścieżka jest folderem czy plikiem. W pierwszym przypadku zwraca „1”, w drugim „0” lub „-1” w przypadku wystąpienia błędu przy odczycie informacji ze ścieżki.*

1. **void fsleep(int sig)**

*Ta funkcja wpisuje log, że demon wznowił działanie po określonym w programie czasie.*

1. **void fsig(int sig)**

*Ta funkcja wpisuje log, że nastąpiło odebranie sygnału SIGUSR1, który powoduje natychmiastowe wznowienie działania demona.*

1. **void Patch\_is\_Dir(char\* path)**

*Ta funkcja sprawdza, czy podana ścieżka jest katalogiem. Jest wykorzystywana przy sprawdzaniu poprawności wprowadzonych argumentów, konkretniej, czy obie ścieżki wskazują na katalogi.*

1. **int copyFile(char \*source, char \*target)**

*Ta funkcja odpowiada za przekopiowanie plików z katalogu źródłowego do katalogu docelowego. Jest wykorzystywana dla plików, których rozmiar nie przekracza rozmiaru granicznego. Wykorzystuje funkcję open do uzyskania danych zawartych w pliku.*

1. **int copyFile\_mmap(char \*source, char \*target, struct stat \*st)**

*Ta funkcja odpowiada za kopiowanie plików z katalogu źródłowego do katalogu docelowego, jeśli rozmiar pliku przekracza rozmiar graniczny. Wykorzystuje do tego mapowanie pliku, co przyspiesza proces kopiowania pliku.*

1. **int synchronize(char \*src, char \*dst, int rec, long int size)**

*Jest to główna funkcja działająca cyklicznie. Jej zadaniem jest analizowanie obu katalogów w celu podjęcia decyzji które pliki należy skopiować, usunąć, zamienić oraz w jaki sposób ma się to odbywać. Funkcja sprawdza każdą pozycję w katalogu źródłowym i jeśli została przekazana opcja „-R” jest synchronizowana cała zawartość. W przeciwnym wypadku procesowi synchronizacji poddawane są pozycje nie będące katalogami. W przypadku, gdy w katalogu docelowym znajdują się pliki, ale w katalogu źródłowym jest ich nowsza wersja, to zostaje on podmieniony w katalogu docelowym na nowszy.*

1. **void RemoveExtraFiles(char \*src, char \*dst, int rec)**

*Jest to funkcja uruchamiana po zakończeniu funkcji synchronize(). Jest ona odpowiedzialna za usuwanie elementów, które występują w katalogu docelowym, ale nie występują w katalogu źródłowym. Zapewnia, że katalog docelowy jest identyczną kopią katalogu źródłowego.*