Temat projektu: Demon synchronizujący dwa podkatalogi
Wykonujący projekt: Szymon Laskowski, Paweł Kamieński (PS10)

W projekcie zaimplementowano wszystkie wymagane funkcjonalności.

1. Porównywanie dwóch podanych katalogów, które są podawane jako argumenty **-s** i **-d** (source i destination)
2. Ustawienie czasu uśpienia demona (demon domyślnie śpi przez 5 minut) jako argument **-n** (nap)
3. W zależności od rozmiaru plików dla małych plików wykonywane jest kopiowanie przy pomocy read/write a w przypadku dużych używany jest bardziej efektywny mechanizm, np.: przy pomocy mmap/write (plik źródłowy zostaje zamapowany w całości w pamięci) lub za pomocą dedykowanego wywołania (np. sendfile czy copy_file_range). Próg dzielący pliki małe od dużych może być przekazywany jako opcjonalny argument **-b**. Domyślny rozmiar to 20.
4. Opcja **-r** pozwalająca na rekurencyjną synchronizację katalogów (teraz pozycje będące katalogami nie są ignorowane). W szczególności jeżeli demon stwierdzi w katalogu docelowym podkatalog którego brak w katalogu źródłowym powinien usunąć go wraz z zawartością.
5. Podczas kopiowania ustawiane są prawidłowe daty modyfikacji.
6. Natychmiastowe obudzenie demona za pomocą wysłania sygnału SIGUSR1 (kill -SIGUSR1 pid)

Wyczerpująca informacja o każdej akcji typu uśpienie/obudzenie się demona (naturalne lub w wyniku sygnału), wykonanie kopii lub usuniecie pliku jest przesłana do logu systemowego. Informacja ta zawiera aktualna datę.

- **sigusr1_handler** (ręczne wybudzenie demona)
- **setModificationTime** (błąd przy zmianie daty modyfikacji)
- **get_size_file** (błąd przy pobieraniu rozmiaru pliku)
- **CheckMtime** (błąd przy zmianie czasu modyfikacji)
- **Delete** (poinformowanie o tym, że usunięto plik wraz nazwą tego pliku)
- **Copy** (informacja o tym, który sposób kopiowania został użyty do zapisania nowego pliku)
- **CopyFile** (Błąd przy otwarciu/kopiowaniu/zapisie pliku)
- **CopyFileWithmapping** (Błąd przy otwarciu/ mapowania oraz gdy niepowodzenie munmap)
- **run_demon** (informacja o rozpoczęciu analizy zawartości ścieżki źródłowej oraz o tym kiedy utworzono nowy katalog w ścieżce docelowej)

Analiza MMAP/WRITE VS READ/WRITE.

Wyniki są zależne od tego jakiego dysku używamy. Znajdując odpowiednie źródło dowiedzieliśmy wielu istotnych cech, które mają wpływ na różnice w wydajności pomiędzy tymi sposobami kopiowania danych.

Mianowicie:

- wersja jądra
- rodzaj dysku
- rozmiar pliku
- plik jest buforowany czy nie
- rozmiar bufora (dla metody READ/WRITE)

PLIK	READ/WRITE	MMAP/WRITE
100MB	0.11s (rozmiar bufora = 32)	0.01s
100MB	0.04s (rozmiar bufora = 128)	0.01s

Czym większy nasz bufor tym nasze pliki, będą kopiowały się szybciej.

void sigusr1_handler(int signal)
Przyjmuje wartość int (sygnał)
Funkcja do ręcznego wybudzania demona poprzez przekazanie sygnału

bool check_file_exist(char* file)
Funkcja przyjmująca ścieżkę do pliku
zwracająca wartość True/1 jeśli plik istnieje, False/0 jeśli nie istnieje

void setModificationTime(char* source_path, char* target_path)
Funkcja przyjmująca 2 ścieżki do plików ustawiająca czas modyfikacji pliku target_path na czas modyfikacji pliku source_path.
Jeśli wystąpi błąd przy zmianie czasu ostatniej modyfikacji wysyła informację do logów

off_t get_size_file(char *source)

Funkcja przyjmująca ścieżkę do pliku i zwracająca rozmiar pliku,
jeśli plik nie istnieje zwraca -1

long comparator_time(char* source, char* target)
Funkcja przyjmująca 2 ścieżki do plików
służąca do porównywania czasów modyfikacji
Zwraca 1 jeśli równe
Zwraca 0 jeśli nierówne

int CheckDirectory(char* dir)
Funkcja pomocnicza do użycia w funkcjach Delete,run_demon
Przyjmuje ścieżkę do pliku
Jeżeli jest katalogiem to zwraca 0 ,w przeciwnym wypadku 1.

void IsDirectory(char* dir)
Przyjmuje ścieżkę do katalogu,
jeśli nie jest katalogiem funkcja powraca z komunikatem błędu.

time_t CheckMtime(char* file)
Funkcja przyjmująca ścieżkę do pliku
zwracająca czas modyfikacji pliku, jeśli nie uda się pobrać wysyła informacje do logów o błędzie

int check_regular_file(char* path)
Funkcja przyjmująca ścieżkę do pliku
Zwracająca wartość 1 jeśli plik jest plikiem regularnym, w przeciwnym wypadku 0

void Delete(char* path_source, char* path_destination, int recursion)
Funkcja przyjmująca 2 ścieżki do plików
oraz wartość rekurencji, potrzebną do zdecydowania czy funkcja Delete ma sprawdzać katalogi czy tylko pliki regularne.
Jeśli int recursion =1 (True) to funkcja sprawdza również katalogi występujące w ścieżce docelowej
Jeśli int recursion = 0 (False) to sprawdza tylko katalog znajdujący się pod przesłaną ściezką path_destination

void Copy(char *source,char *target,int size_bufor)
funkcja przyjmująca dwie zmienne typu char oraz rozmiar bufora
Funkcja decyduje, którego kopiowania użyć
Jeśli rozmiar pliku jest mniejszy równy od size_bufor to użyje funkcji **CopyFile** w innym przypadku użyje funkcji **CopyFileWithMapping**

void CopyFile (char *input,char *output , int size_of_bufor)
funkcja przyjmująca dwie zmienne typu char oraz rozmiar bufora korzystająca z niskopoziomowych read/write

//http://man7.org/linux/man-pages/man2/mmap.2.html
void CopyFileWithMapping (char *input, char *output)
funkcja przyjmująca dwie zmienne typu char - ścieżki do plików, które mają zostać skopiowane z char *input do char *output
Funkcja wykorzystuje mmap, które kopiuje dane z obszaru kopiowanego do oddzielnego bufora “docelowego”

char* create_file_path(char* path,char* addition)
Funkcja przyjmująca (paweł dokończ)
Funkcja zwracająca całą ścieżkę do pliku

void run_demon(char* path_source, char* path_destination, int recursion, int bufor_size)
Funkcja uruchamiająca Demona