**Dokumentacja Projektu**

Temat projektu: **1. Demon synchronizujący dwa podkatalogi**

Autorzy: Pernal Mateusz, Rosiński Damian

Grupa: PS4

Wykorzystane **biblioteki** w projekcie:

* #include <stdio.h>
* #include <sys/stat.h>
* #include <string.h>
* #include <stdlib.h>
* #include "function.h"
* #include <fcntl.h>
* #include <syslog.h>
* #include <unistd.h>
* #include <sys/mman.h>
* #include <dirent.h>

Wykorzystane **pliki** w projekcie:

* main.c – zawiera kod programu.
* function.h – zawiera nagłówki funkcji, które zostały wykorzystane w programie.
* function.c – zawiera ciała funkcji wykorzystanych w projekcie.

Opis poszczególnych funkcji, zasada ich działania.

**help()** – Procedura wykorzystywana w sytuacjach, gdy użytkownik błędnie uruchomi program i wprowadzi niepoprawnie argumenty początkowe.

**handler() –** procedura, która wprowadza nową funkcjonalność do obsługi sygnału.

**if\_file\_handler\_for\_fun\_comp() –** procedura sprawdzająca warunki czy istnieje dany plik, zawiera obsługę kopiowania zwykłych plików, zawiera procedurę **start\_copy()** , która jest odpowiedzialna za część synchronizacji, która kopiuje pliki z katalogu źródłowego do docelowego: (a) w przypadku gdy data modyfikacji w katalogu docelowym jest późniejsza niż w docelowym oraz (b) gdy brakuje pliku z katalogu źródłowego w katalogu docelowym. Kopiowanie odbywa się za pomocą dwóch funkcji **copy\_read\_write()** lub **copy\_nmap()**.

**copy\_read\_write()** – funkcja, odpowiedzialna za proces kopiowania plików, gdy rozmiar kopiowanego pliku jest mniejszy od podanej wartości przy parametrze *–S,* ewentualnie jego wartości domyślnej.

**copy\_nmap()** – funkcja, wywoływana podczas kopiowania pliku, która kopiuje go za pomocą mapowania. Wywoływana jest, gdy rozmiar kopiowanego pliku jest większy od podanej wartości przy parametrze *–S,* ewentualnie jego wartości domyślnej.

**read\_args()** – funkcja, służy do sprawdzenia argumentów startowych uruchamianego demona. W niej zawarta jest procedura **is\_good\_path()**, która sprawdza czy została podana prawidłowa ilość argumentów do poprawnego uruchomienia programu, czy pierwszy i drugi argument jest katalogiem, oraz procedura **check\_params()** w niej zawarta weryfikuje opcjonalne argumenty i ich wartości.

**default\_value\_set()** – Procedura, która w przypadku braku podania opcjonalnych parametrów wejściowych i ich wartości demona ustawia wartości domyślne.

**is\_regular\_file()** – funkcja, służy do określania czy podana ścieżka do pliku jest zwykłym plikiem.

**void signal\_change\_fun()** – Funkcja zmieniająca domyślny mechanizm obsługi sygnału SIGUSR1, który może być zdefiniowany przez użytkownika. U nas zdefiniowana jest jako obudzenie demona.

**start\_new\_procces()** – Funkcja tworząca nowy proces potomny, tak aby demon mógł pracować w tle, niezależnie od proces macierzystego.

**add\_path()** - funkcja, która konkatenuje ze sobą dwa napisy. Zazwyczaj służy do tworzenia ścieżki do konkretnego pliku, który jest sprawdzany lub ew. modyfikowany. Np.: *input*=”/root/Projekt/Plik\_źródłowy” , *what\_to\_do* zazwyczaj przyjmuje nazwę konkretnego pliku np. *what\_to\_do=”tekstowy1”,* żeby stworzyć ścieżkę do pliku ”/root/Projekt/Plik\_źródłowy/tekstowy1”.

**compare\_files()** – funkcja, służy do synchronizacji plików przez kopiowanie lub tworzenie. Otrzymuje jako pierwszy argument ścieżkę do pliku źródłowego, a jako drugi ścieżkę do pliku, który będzie synchronizowany. Wykorzystuje ona strukturę *dirent,* dzięki której możliwe jest operowanie na zawartości pliku oraz jego atrybutach. Uwzględnione zostały w niej katalogi specjalne takie jak ‘**.**’ oraz ‘**..**’, które nie są synchronizowane.

Synchronizuje ona każdy plik z osobna przechowując jego ścieżkę z pliku źródłowego w zmiennej *source\_file\_path* oraz ścieżkę tego samego pliku do synchronizowania znajdującego się w katalogu docelowym w zmiennej *destination\_file\_path* jeżeli istnieje.

Ścieżka do każdego pliku z osobna trafia jako argument do funkcji *stat()* w celu możliwości uzyskania statusu tego pliku oraz korzystania z jego atrybutów takich jak: typ pliku, rozmiar pliku, prawa dostępu, czy czas dostępu.

Korzysta z procedury **if\_file\_handler\_for\_fun\_comp(),** która obsługuje pierwszy wariant synchronizacji, gdy program został uruchomiony **bez parametru** **-R.** Wtedy synchronizowane są jedynie zwykłe pliki, jeżeli data ich modyfikacji w pliku docelowym jest późniejsza lub nie istnieją one w pliku docelowym. Ich synchronizacja następuje przez kopiowanie lub stworzenie. Natomiast katalogi są pomijane.

W przypadku , gdy program został uruchomiony **z parametrem** **–R,** korzysta z procedury **if\_is\_directory\_handler\_for\_fun\_comp(),** która tworzy lub kopiuje katalog. Przy czym zwykłe pliki są nadal kopiowane przy pomocy **if\_file\_handler\_for\_fun\_comp().** Wtedy synchronizowane są wszystkie pliki włącznie z katalogami w sposób rekurencyjny, jeżeli data ich modyfikacji w pliku docelowym jest późniejsza lub nie istnieją one w pliku docelowym. Ich synchronizacja następuje przez kopiowanie w przypadku plików tekstowych lub stworzenie w przypadku katalogów.

**deleting\_files()** - funkcja, służy do synchronizacji katalogów przez usuwanie nadmiernych plików w katalogu docelowym. Otrzymuje jako pierwszy argument ścieżkę do pliku źródłowego, a jako drugi ścieżkę do pliku, który będzie synchronizowany. Wykorzystuje ona strukturę *dirent,* dzięki której możliwe jest operowanie na zawartości pliku oraz jego atrybutach. Uwzględnione zostały w niej katalogi specjalne takie jak ‘**.**’ oraz ‘**..**’, które nie są synchronizowane.

Przed synchronizacją pliku docelowego program sprawdza, czy jest to katalog. Jeżeli nie jest następuje błąd programu i odpowiednia informacja trafia do logów.

Wykorzystana została procedura **if\_file\_handler\_for\_deleting()**, która obsługuje usuwanie zwykłych plików,sprawdza jedynie czy w pliku docelowym i źródłowym występują zwykłe pliki o takiej samej nazwie. Jeżeli w katalogu źródłowym brakuje takiego pliku, jest on usuwany z katalogu docelowego.

W przypadku kiedy program synchronizuje katalogi przez usuwanie z **włączonym parametrem –R**, korzystna z procedury **if\_directory\_handler\_for\_deleting()**, obsługującej usuwanie katalogów. Sprawdza czy plik docelowy jest katalogiem, zarówno czy plik o tej samej nazwie w katalogu źródłowym jest katalogiem i czy istnieje. Jeżeli tak, to program nie wprowadza zmian, natomiast gdy brakuje tego katalogu w pliku źródłowym, jest on usuwany.