Katedra Oprogramowania **Systemy operacyjne**	Data rozpoczęcia projektu: 24.03.2021r Data oddania projektu: 12.04.2021r.
Temat projektu: **Demon synchronizujący dwa katalogi** Skład grupy projektowej: 1. Arkadiusz Abramowicz 2. Magdalena Korecka	Prowadzący: dr inż. Marcin Koźniewski Ocena: ……..

1. Treść projektu do realizacji:

Program który otrzymuje co najmniej dwa argumenty: ścieżkę źródłową, ścieżkę docelowa . Jeżeli któraś ze ścieżek nie jest katalogiem program powraca natychmiast z komunikatem błędu. W przeciwnym wypadku staje się demonem. Demon wykonuje następujące czynności: śpi przez piec minut (czas spania można zmieniać przy pomocy dodatkowego opcjonalnego argumentu), po czym po obudzeniu się porównuje katalog źródłowy z katalogiem docelowym. Pozycje, które nie są zwykłymi plikami są ignorowane (np. katalogi i dowiązania symboliczne). Jeżeli demon (a) napotka na nowy plik w katalogu źródłowym, i tego pliku brak w katalogu docelowym lub (b) plik w katalogu docelowym ma późniejsza datę ostatniej modyfikacji demon wykonuje kopie pliku z katalogu źródłowego do katalogu docelowego - ustawiając w katalogu docelowym datę modyfikacji tak, aby przy kolejnym obudzeniu nie trzeba było wykonać kopii (chyba ze plik w katalogu źródłowym zostanie ponownie zmieniony). Jeżeli zaś odnajdzie plik w katalogu docelowym, którego nie ma w katalogu źródłowym to usuwa ten plik z katalogu docelowego. Możliwe jest również natychmiastowe obudzenie się demona poprzez wysłanie mu sygnału SIGUSR1. 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 powinna zawierać aktualna datę. Operacje kopiowania mają być wykonane za pomocą niskopoziomowych operacji read/write. (14p). Dodatkowo:

- 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. (8p)

-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. Wykonaj analizę wpływu danej metody na szybkość kopiowania plików i przedstaw wyniki w dokumentacji. (12p)

2. Sposób uruchomienia demona:

Archiwum o nazwie „demon_synchronizujący.zip” należy rozpakować oraz przejść do folderu z wypakowanymi plikami projektowymi. W otwartym folderze należy otworzyć terminal i za pierwszym razem wpisać komendę ***make,*** w celu skompilowania projektu. Po pojawieniu się komunikatu ***gcc main.c projectfunctions.c -o projekt*** program został skompilowany bez błędów i jest możliwe uruchomienie projektu.

2.1. Sposoby w jaki sposób można uruchomić program.

1. Parametry wymagane:

Komendą ***./projekt -s [sciezka_zrodlowa] -d [sciezka_docelowa]*** można uruchomić program w podstawowy sposób. Parametry:

-s [sciezka_zrodlowa] – podanie dla przyszłego demona ścieżki katalogu źródłowego, skąd ma kopiować pliki do drugiego katalogu,
-d [sciezka_docelowa] - podanie dla przyszłego demona ścieżki katalogu docelowego, dokąd ma kopiować pliki z katalogu źródłowego lub usuwać pliki, które nie znalazły się w katalogu źródłowym.

UWAGA!!!
Katalogi muszą istnieć, aby program działał poprawnie oraz ścieżka źródłowa i docelowa nie może być taka sama.

2. Parametry opcjonalne:

-R – pozwala na rekurencyjne sprawdzenie katalogów.
-T [czas w sekundach] – ustawienie czasu, po którym demon na zacząć synchronizację,
-S [rozmiar] – ustawienie rozmiaru, po którym demon ma sprawdzać, którą funkcję kopiującą ma użyć. Bez użycia tego parametru wszystkie pliki są kopiowane za pomocą niskopoziomowych funkcji read/write. Pliki wyższej niż parametr **rozmiar** będą kopiowane za pomocą dedykowanego wywołania sendfile.
-h – wyświetlenie na standardowe wyjście wszystkich argumentów, jakie program może przyjąć. Po uzyskaniu parametru **-h** program zamyka się.

Przykładowe uruchomienia:

./projekt -s /home/folder1/zrodlo -d /home/foo/23/docelowy -R -T 20 -S 800

./projekt -d /home/ffo -s /home/folder1/f12345 -S 20

3. Sposób testowania projektu:

Aby przetestować projekt należy użyć odpowiednio ./skrypt[nr_skryptu] po rozpakowaniu i skompilowaniu projektu. Następnie trzeba przejść do logów systemowych i obserwować zmiany.

Polecenia do testowania:

./projekt -s /home/arke/proba1/kat1/folder1 *-d* /home/arke/proba1/kat1/folder2 *-R -T 20*

./projekt -s /home/arke/proba1/kat2/folder1 *-d* /home/arke/proba1/kat2/folder2 *-T 20 -S 1*

4. Opis zaimplementowanych funkcji:

int addSlash(char *entry_path, int path_len) – funkcja dodatkowa, służąca do sprawdzenia, czy ścieżka **entry_path** do folderu zakończona jest znakiem „/”, jeżeli nie to dodaje znak na sam koniec. Zwraca wartości 1 gdy dodało znak „/” oraz 0 gdy był znak na końcu ścieżki.

void bigFileCopyingFunction(char *entry_path_source, char *entry_path_destination) – funkcja służąca do kopiowania większych plików ze ścieżki źródłowej **entry_path_source** do ścieżki docelowej **entry_path_destination** za pomocą funkcji **sendfile.**

void smallFilecopyingFunction(char *entry_path_source, char *entry_path_destination) - funkcja służąca do kopiowania większych plików ze ścieżki źródłowej **entry_path_source** do ścieżki docelowej **entry_path_destination** za pomocą funkcji niskopoziomowych **read/write.**

void copyingFunction(char *entry_path_source, char *entry_path_destination, int size) – funkcja sprawdzająca, czy plik jest większy od zadanego rozmiaru odróżniającego pliki duże od małych. Jeżeli jest większy rozmiar pliku od **size**, to zostaje wywołana funkcja **bigFileCopyingFunction,** jeżeli mniejsza - **smallFilecopyingFunction.**

void deletingFunction(char *entry_path_destination) – funkcja usuwająca dany plik o ścieżce **entry_path_destination**. Jeżeli to jest folder, w którym są pliki to usuwa wszystkie pliki i na sam koniec usuwa dany folder.

void browsingTheDirectories(char *source, char *destination, int recursion, int size) – główna funkcja demona, która pobiera listę plików z folderu źródłowego **source** oraz z folderu docelowego **destination** uporządkowane alfabetycznie i sprawdza odpowiednio pliki:

- jeżeli folder docelowy jest pusty to ma przekopiować wszystkie pliki.
- w pętli przechodzi po wszystkich plikach folderu zrodlowego i porownuje je z plikami z folderu docelowego:

Porównania plików są odpowiednie, gdyż wynika to z faktu, iż są posortowane alfabetycznie.

- jeżeli nazwa pliku z folderu źródłowego jest **wieksza** niż z folderu docelowego to ma usunąć plik z folderu docelowego, przykład:

Folder1:	Folder2:
-ab	-aa
-aj	-ab

-aj

- jeżeli nazwa pliku z folderu źródłowego jest **mniejsza** niż z folderu docelowego to ma skopiować plik do folderu docelowego, przykład:

Folder1:	Folder2:
-ab	-aj
-aj	

- jeżeli nazwa pliku z folderu źródłowego jest **taka sama** jak z folderu docelowego to musi demon musi porównać datę modyfikacji. Jeżeli się różni to usuwa dany plik z folderu docelowego i kopiuje go do folderu docelowego.

void man_projekt_program() - funkcja dodatkowa, wyświetlająca na standardowe wyjście pomoc dla użytkownika programu. Jest wywoływana po przekazaniu przy starcie programu parametru **-h**.