Politechnika Białostocka

Dokumentacja projektu

Systemy Operacyjne

Projekt numer 1

Temat:**Demon synchronizujący dwa podkatalogi**

Wykonujący: **Paweł Halicki, Michał Kierzkowski**

Studia dzienne

Kierunek: Informatyka

Semestr: IV

Grupa zajęciowa: PS1

Prowadzący ćwiczenie: **mgr inż. Mirosław Marzewski**

18.04.2018
Data oddania projektu

…….....
Data i podpis prowadzącego

TREŚĆ PROJEKTU

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.

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.
- W zależności od rozmiaru plików dla małych plików wykonywane jest kopiowanie przy pomocy read/write a w przypadku dużych przy pomocy mmap/write (plik źródłowy) zostaje zmapowany w całości w pamięci. Próg dzielący pliki małe od dużych może być przekazywany jako opcjonalny argument.

LISTA PLIKÓW

Pliki z kodem źródłowym daemon:	main.cpp
PlikMakefile:	makefile
Plik wykonywalny (skompilowany projekt):	Daemon

LISTA FUNKCJONALOŚCI

- ✓ Dwa tryby pracy: iteracyjny (synchronizacja z ignorowaniem katalogów) i rekurencyjny (synchronizacja wraz z katalogami).
- ✓ Synchronizowanie, co dany interwał czasy (domyślnie pięć minut z możliwością zmiany za pomocą argumentu).
- ✓ W zależności od rozmiaru pliku synchronizacja odbywa się dla małych plików przy pomocy read/write a w przypadku dużych przy pomocy mmap/write (granica dzieląca pliki małe od dużych został ustawiona na poziomie 1MB).
- ✓ Możliwość natychmiastowego uruchomienia demona poprzez wysłanie mu sygnału SIGUSR1.
- ✓ Informacja o każdej akcji typu uśpienie/obudzenie się demona, wykonanie kopii lub usuniecie pliku jest przesłana do logu systemowego (informacja zawiera aktualną datę).

LISTA FUNKCJI

Funkcje API systemu Linux:

- chmod() –fukcja służąca do zmiany praw dostępu do pliku w systemach
- stat() – funkcja zwracająca inforamcje o pliku
- readdir() –funkcja zwraca wskaźnik do nowej struktury reprezentującej następny wpis katalogu w strumieniu katalogów wskazywany przez dirp
- mkdir() –funkcja służąca do tworzenia nowych katalogów w systemie
- opendir() –funkcja otwierająca strumień katalogu odpowiadający nazwie katalogu i zwracająca wskaźnik do strumienia katalogu
- rmdir() – funkcja służąca do usuwania pustych katalogów
- open() –funkcja otwierająca plik. Zwraca deskryptor pliku(nieujemna liczba uzywana przy czytaniu, pisaniu do pliku, itd.) lub -1 w przypadku błędu(kod bledu na zmiennej errno)
- write()–funkcja służąca do pisania do pliku
- close() –funkcja zwalnia deskryptor pliku wskazany przez fildes
- perror() – funkcja generuje komunikat na standardowym wyjściu błędu, opisujący ostatni błąd napotkany podczas wywołania funkcji systemowej lub bibliotecznej
- syslog() – funkcja służąca generowaniu komunikatu dziennika, który będzie dystrybuowany przez logi systemu
- mmap() –funkcja pozwalająca na odwzorowanie wybranego pliku w przestrzeni adresowej procesu, dzięki niej do obszaru pliku możemy odnieść się jak do zwykłej tablicy bajtów w pamięci.
- memcpy() –funkcja kopiująca n bajtów z obszaru pamięci src do obszaru pamięci dest.
- munmap() –funkcja usuwająca wszelkie mapowania dla całych stron zawierających dowolną część przestrzeni adresowej procesu zaczynając od adresu addr i kontynuując dla n bajtów
- unlink() – funkcja służąca do usuwania pojedynczego pliku, jednak w przeciwieństwie do polecenia rm usuwa jedynie dowiązanie do nazwy, nie oznacza miejsca na partycji jako wolne
- asctime() – funkcja służąca do uzyskania złożonego łańcucha zawierającego pełną datę
- localtime() –funkcja zwracająca dane reprezentujące czas kalendarzowy
- signal() –funkcja do obsługi sygnału
- sleep() –funkcja sprawia, że wątek wywołujący pozostaje uśpiony przez daną ilość sekund lub do czasu wysłania sygnału, który nie jest ignorowany.
- exit() – funkcja powodując prawidłowe zakończenie procesu

Funkcje własne:

- main()-funkcja main. Wywołuje poniższe funkcje w zależności od tego co zostało przekazane przy uruchamianiu demona.W zależności od tego jakie argumenty zostały podane na wejściu następuje ewentualna zmiana domyślnych ustawień programu np. czas uśpienia, tryb rekurencyjny.Dodana została funkcja manualnego wywoływania synchronizacji danych w przypadku złapania sygnału SIGUSR1.
- voidsig_handler()-funkcja służąca do wysyłania sygnału zdefiniowanego przez użytkownika (SIGUSR1), sygnał wykorzystywany jest do manualnego wzbudzenia synchronizowania danych
- intsync()- funkcja wykorzystywana do wybierania trybu, dzięki któremu nastąpi kopiowanie (iteracyjnym lub rekurencyjnym)
- intsyncFlat()- funkcja synchronizująca dwa katalogi w sposób iteracyjny
- boolcompareTimespec() - funkcja porównująca czasy modyfikacji plików z katalogu źródłowym i docelowym
- intcopyFile() - funkcja wykorzystywana do wybierania sposobu, dzięki któremu nastąpi kopiowanie (w zależności od wielkości pliku) regularne lub przez mapownie.
- intcopyFileIO()– tryb normalnego kopiowania za pomocą standardowych funkcji read/write – zalecana dla małych plików
- intcopyFileByMmap()– tryb kopiowania z wykorzystaniem mapowania – zalecana dla większych plików
- intforceRemove() – funkcja mająca na celu ustalenie czy podana ścieżka odnosi się do pliku czy katalogu a następnie jego usunięcie
- intforceRemoveDir() - funkcja mająca na celu usunięcie katalogu
- intisDir() - funkcja mająca na celu sprawdzenie czy element podany na wejściu jest katalogiem.
- intisRegularFile() - funkcja mająca na celu sprawdzenie czy element podany na wejściu jest plikem.
- structtm* getTime() -funkcja zwracająca aktualny czas

SPOSÓB URUCHOMIENIA

1) Kompilacja programu w systemie.
2) Uruchomienie plik wykonywalnego:
    a. Iteracyjnie – z parametrami: folder źródłowy, folder docelowy, oraz opcjonalnie: czas pomiędzy synchronizacjami (sekundy)
    Np. ./daemon /Home/user/exampleSourceDir /home/user/exampleDestinationDir300

b. Rekurencyjne – z parametrami: -R,folder źródłowy, folder docelowy, oraz opcjonalnie: czas pomiędzy synchronizacjami (sekundy)
Np. ./daemon –R /home/user/exampleSourceDir /home/user/exampleDestinationDir300

3) Po uruchomieniu pliku wykonywalnego istnieje możliwość natychmiastowej synchronizacji (bez potrzeby odczekiwania danego interwału czasowego) przy pomocy wysłania sygnału SIGUSR1. Np. kill-SIGUSR1 4040

TESTOWANIE

- Tworzymy dwa katalogi: źródłowy i docelowy, następnie w katalogu źródłowym tworzymy losowe pliki.
- Kompilujemy plik wykonywalny, a następnie uruchomić go w odpowiednim dla testu trybie pracy wraz ze ścieżkami do wcześniej utworzonych plików
- Aby sprawdzić działanie natychmiastowej synchronizacji wysyłamy sygnał SIGUSR1
- Podczas testowania zmieniamy zawartość katalogu źródłowego tak aby przetestować różne możliwości demona (takie jak mapowanie).
- Przechodzimy do logów systemowych (za pomocą cat /var/log/syslog), aby sprawdzić czy nasz program przesłał tam odpowiednie informacje o każdej akcji.
- Dezaktywujemy demona.