Wydział Informatyki Politechniki Białostockiej Przedmiot: Systemy Operacyjne	Data: 05.05.2022
Temat: Projekt 1 Grupa: PS 1 Imię i nazwisko: **Szymon Siłkin** **Paweł Siemieniuk** **Marcel Herman**	Prowadzący: **dr. inż. Wojciech Kwedlo**

Temat 2 - Demon synchronizujący dwa podkatalogi

[12p.] Program który otrzymuje co najmniej dwa argumenty: ścieżkę źródłową oraz ścieżkę docelową . 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 pięć 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 źrodłowym ma późniejszą datę ostatniej modyfikacji demon wykonuje kopię 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 usunięcie pliku jest przesłana do logu systemowego. Informacja ta powinna zawierać aktualną datę.

[10*p.*] Dodatkowa 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.

[12*p.*] 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 zamapowany w całości w pamięci. Próg dzielący pliki małe od dużych może być przekazywany jako opcjonalny argument.

Uwagi: (a) Wszelkie operacje na plikach i tworzenie demona należy wykonywać przy pomocy API Linuksa a nie standardowej biblioteki języka C (b) kopiowanie za każdym obudzeniem całego drzewa katalogów zostanie potraktowane jako poważny błąd (c) podobnie jak przerzucenie części zadań na shell systemowy (funkcja system).

Realizacja punktów

Zrealizowane

+ Synchronizacja plików w dostarczonych katalogach oraz ich podkatalogach.
+ Wychwytywanie błędów związanych z podanymi argumentami.
+ Dostosowywalny czas wstrzymania programu (snu demona).
+ Obsługa sygnału SIGUSR1.
+ Rejestrowanie każdej akcji i wysyłanie jej do logu.
+ Dwa sposoby kopiowania w zależności od wielkości pliku.

Niezrealizowane

- Puste katalogi nie są kopiowane ani usuwane.
- Program nie przewiduje usunięcia bazowych katalogów (podanych w argumentach).

Najistotniejsze algorytmy

Wczytywanie plików do listy

Funkcja wczytuje kolejne znalezione pliki na listę do momentu gdy natrafi na katalog (opcja -R). Po natrafieniu na niego, wywołuje się rekurencyjnie zmieniając ścieżkę skanowanego katalogu na tego napotkanego. Po dodaniu do listy plików z podkatalogu, powraca do wczytywania kolejnych plików.

Fragment dir_op.c

```
void readDir(f_list **list, char *pathname)
{
        DIR *dir = opendir(pathname);
        struct dirent *read_file;
        while((read_file = readdir(dir)) != NULL)
        {
                unsigned char file_type = read_file->d_type;
                char *filename = read_file->d_name;
                char *file_path = calloc(strlen(pathname) + strlen(filename) +
                    2, sizeof(char));
                strcat(file_path, pathname);
                strcat(file_path, "/");
                strcat(file_path, filename);

                struct stat *file_buff = calloc(1, sizeof(struct stat));
                lstat(file_path, file_buff);

                if(file_type == DT_REG){
                        (*list) = push((*list),
                                        pathname,
                                        filename,
                                        file_buff->st_size,
                                        file_buff->st_mtime);
                }
                else if(F_SUBDIR && strcmp(filename, "..") && strcmp(filename,
                    ".") && file_type == DT_DIR){
                        long dir_loc = telldir(dir);
                        closedir(dir);

                        readDir(list, file_path);

                        dir = opendir(pathname);
                        seekdir(dir, dir_loc);
                }
                free(file_path);
                free(file_buff);
        }
        closedir(dir);
}
```

Zmienne Globalne i Funkcje

Zmienne Globalne

Plik var.h

```c
#include <sys/types.h>
#include <time.h>
#include <stdbool.h>

#ifndef VAR
#define VAR

// Sciezki bezwzgledne katalogu zrodlowego i docelowego
extern char *SRC_NAME, *DST_NAME;

// Flaga -R przelaczajaca sprawdzanie podkatalogow
extern bool F_SUBDIR;

// Rozmiar pliku po ktorym zostanie on zapisany przy pomocy mmap()
extern unsigned int BIG_FILE_SIZE;

// Czas po ktorym nastapi ponowna synchronizacja
extern unsigned int SLEEP_TIME;

// Informacje o pliku
typedef struct file_info {
        char    f_name[256];    // Nazwa pliku
        off_t   f_size;         // Rozmiar pliku w bajtach
        time_t  f_mtime;        // Data ostatniej modyfikacji w sekundach
} f_info;

// Struktura elementu listy plikow
typedef struct file_list {
        bool  checked;          // Czy plik w porownywaniu zostal sprawdzony
        char *path;             // Sciezka do pliku
        struct f_info *file_i;

        struct f_list *next;
} f_list;
#endif
```

Logi

Plik log.h

```
#ifndef LOG
#define LOG

// Zapis wykonanych akcji w logs/actions.log
void logAction(char *action);

#endif
```

Operacje związane z katalogami i ścieżkami

Plik dir_op.h

```
#include "var.h"

#ifndef DIR_OP
#define DIR_OP

// Wczytywanie plikow ze sciezki do listy
void readDir(f_list **list, char *pathname);

// Kopiowanie plikow na liscie z katalogu zrodlowego do katalogu docelowego
void copyDir(f_list **src_list);

// Usuwanie zbednych plikow z katalogu docelowego
void cleanDir(f_list **dst_list);

#endif
```

Operacje związane z plikami

Plik file_op.h

```
#include "var.h"

#ifndef FILE_OP
#define FILE_OP

// Porownywanie listy plikow z katalogu zrodlowego i docelowego
void fileListCompare(f_list **src_list, f_list **dst_list);

// Porownywanie dwoch plikow
bool fileCompare(f_info *src_file, f_info *dst_file);

// Kopiowanie pliku z katalogu zrodlowego do docelowego
void copyFile(char *path, f_info *file_i);

// Usuwanie pliku
void delFile(char *pathname);

// Usuwanie katalogu
void delDir(char *pathname);

#endif
```

Kopiowanie plików

Plik file_cpy.h

```
#include "var.h"

#ifndef CPY
#define CPY

// Kopiowanie plikow poprzez mapowanie ich w pamieci
void copyMap(char *src_path, f_info *finf);

// Kopiowanie plikow wykorzystujac buffer
void copyNormal(char *src_path, f_info *finf);
#endif
```

Operacje na listach

Plik list_op.h

```
#include "var.h"

#ifndef LIST_OP
#define LIST_OP

// Dodawanie elementu na koniec listy
f_list *push(f_list *list_head, char *path, char *name, off_t size, time_t ↩
    mod_time);

// Czyszczenie listy
void clean(f_list *list_head);

#endif
```

Obsługa sygnału oraz funkcja wątku

Fragment pliku daemon.c

```
// Wskaznik na strukture przechowujaca informacje o watku
pthread_t *bed_t;

// Obsluga sygnalu
void signalHandler(int sig) { if(sig == SIGUSR1) pthread_cancel(*bed_t); }

// Funkcja zatrzymujaca dzialanie programu wywolywana jako watek
void *bedThread() { sleep(SLEEP_TIME); }
```

Instrukcja

Kompilacja

Program jest kompilowany przy pomocy narzędzia make następującą komendą:

```
make build
```

Skompilowany program trafia do katalogu /bin.

Uruchamianie

Program uruchamiany jest w tle komendą:

```
./bin/daemon [ OPCJE ] KATALOG_ZRODLOWY KATALOG_DOCELOWY&
```

KATALOG_ZRODLOWY – Ścieżka do katalogu z którego będą kopiowane pliki

KATALOG_DOCELOWY – Ścieżka do katalogu do którego będą kopiowane pliki

OPCJE

[-t T] – Ustawienie czasu snu demona na T sekund.

[-s S] – Ustawienie granicy rozmiaru S w bajtach, dla której większe pliki mają być kopiowane przy pomocy mapowania.

[-R] – Włączenie opcji sprawdzania podkatalogów.

Przykład użycia:

```
./bin/daemon -t 30 -s 100 -R ./katalog1 ./katalog2&
```

Interakcja

Przedwczesne obudzenie się demona można wywołać przy pomocy skryptu:

```
./wakeup.sh
```

Zakończenie działania programu wykonuje następujący skrypt:

```
./kill.sh
```