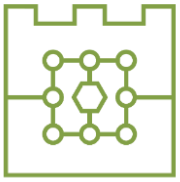
Politechnika Krakowska
im. Tadeusza Kościuszki
Wydział Informatyki i Telekomunikacji



Projekt Cegielnia

Autor:
Jakub Makuch
grupa GL03

Programy

main.c

program główny, obsługujący taśmę cegielni na której stawiane są cegły.
odbiera cegły od pracowników(worker)
kiedy ma pełno cegieł wysyła ciężarówki (truck)

worker.c

wysyła cegły do cegielni(main)
wysyła je przy pomocy kolejki komunikatów - msgsnd

truck.c

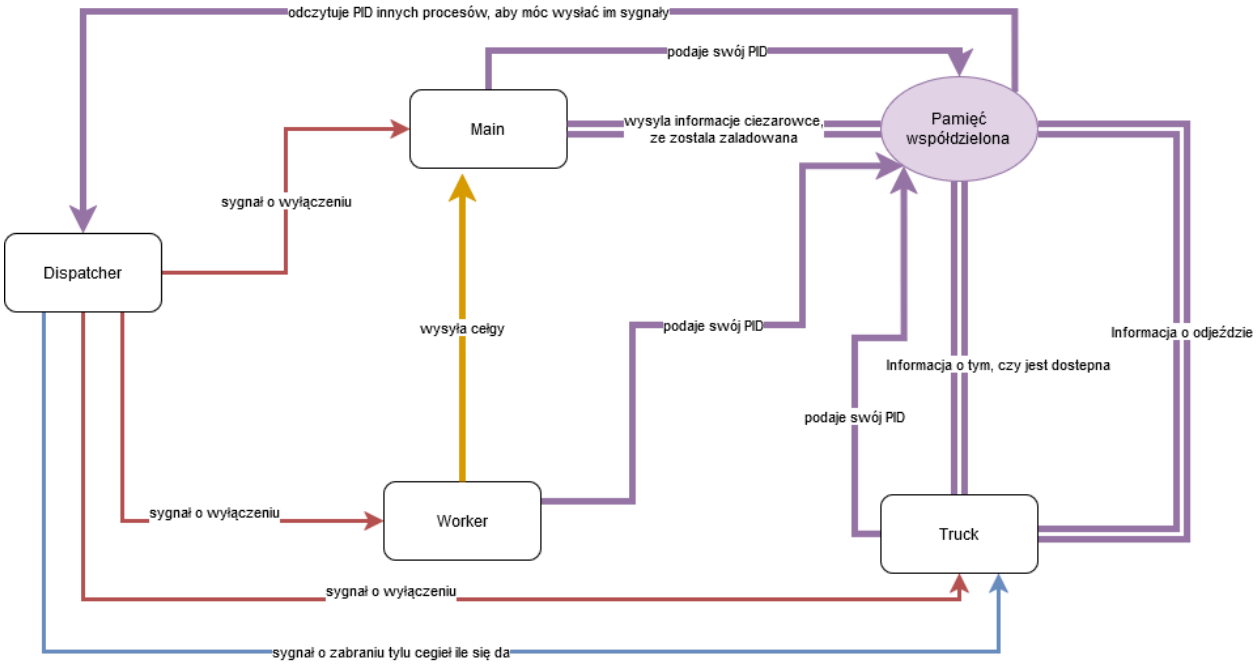
odbiera cegły od cegielni(main)
sprawdza czy zostały mu przydzielone cegły przy pomocy pamięci współdzielonej

dispatcher.c

wysyła sygnały SIGTERM oraz SIGINT do wskazanych procesów.

memory.h

segment na wspólne elementy programu



czerwona strzałka - SIGTERM
niebieska strzałka - SIGINT
żółta strzałka - kolejka komunikatów (wiadomość)
fioletowe - używanie pamięci dzielonej

Użyte struktury IPC

- **kolejki komunikatów**
- **pamięć dzielona**
- **sygnały**

1. **cegielnia (main)**
    a. [link] osobny wątek na odczytywanie komunikatów z kolejki od pracowników
    b. [link] osobny wątek na wyświetlanie stanu cegielni oraz na obsługę ciężarówek
    c. [link] wyslanieCiezarowek(), wykorzystuje pamięć współdzielona do wysyłania ciężarówek
    d. [link] Obsługa sygnału SIGTERM
2. **pracownik (worker)**
    a. [link] korzysta z kolejki komunikatów do wysyłania cegieł do cegielni
        i. wysyłanie komunikatów, wysyłka cegły do cegielni
    b. [link] Obsługa sygnału SIGTERM
3. **ciężarówka (truck)**
    a. [link] pętla while, obsługa logiki przy pomocy pamięci współdzielonej
    b. [link] Obsługa sygnału SIGINT
        i. ten sygnał ustawia ciężarówkę w tryb "pobrania cegieł od razu"
        ii. oraz obsługa sygnału SIGTERM, który kończy program
4. **dispatcher (main)**
    a. [link] Wykorzystuje pamięć współdzieloną do odczytywania PID procesów, które rejestrują swój PID przy rozruchu i wysyła sygnały do poszczególnych procesów na polecenie.
    b. [link] funkcja SendSignal Wysyłają sygnały do poszczególnych procesów.

Opis zgodności z tematem

Jest trzech pracowników oraz dwie ciężarówki. Taśma jest tablicą przyjmującą cegły. Dzięki temu możliwa jest obsługa maksymalnej wagi jak i maksymalnej ilości cegieł. Cegły z taśmy są pobierane w kolejności w jakiej zostały z niej pobrane. Odbieranie cegieł, ściąga się je w kolejności, tablica służy nam tutaj jako kolejka. Pracownicy podają cegielni cegły o wadze 1,2 lub 3. Za każdym razem kiedy na taśmie ląduje cegła, sprawdzane jest czy można wysłać ciężarówkę. Dzięki temu ciężarówka zabiera cegły od razu, kiedy jest w stanie. Informacja o tym, że ciężarówka odjeżdża na określony czas jest obsługiwana w programie truck. Wysyła sygnały SIGTERM do poszczególnych procesów. Jedna z opcji jest ustawienie ciężarówkom flagi, aby pobrały cegły natychmiastowo (SIGINT).

Testy

- Test Worker 1 [link]
    - sprawdza uruchomienie Pracownika przy poprawnych wartościach
    - symulacja komendy: `./worker 1`
- Test Worker 2 [link]
    - sprawdza uruchomienie Pracownika przy niepoprawnych wartościach
    - symulacja komendy: `./worker 99`
- Test Worker 3 [link]
    - sprawdza czy pracownik wysyła kolejkę komunikatów.
    - symulacja komendy: `./worker 2`
- Test Truck 1 [link]
    - sprawdza uruchomienie Ciężarówki przy niepoprawnych wartościach (brak argumentów)
    - symulacja komendy: `./truck`
- Test Truck 2 [link]
    - sprawdza uruchomienie Ciężarówki przy niepoprawnych wartościach (niepoprawne ID)
    - symulacja komendy: `./truck 3 10 10`
- Test Truck 3 [link]
    - sprawdza uruchomienie Ciężarówki przy niepoprawnych wartościach (niepoprawna ładowność)
    - symulacja komendy: `./truck 1 -10 10`
- Test Truck 4 [link]
    - sprawdza uruchomienie Ciężarówki przy niepoprawnych wartościach (niepoprawny czas jazdy)
    - symulacja komendy: `./truck 1 100 -5`
- Test Truck 5 [link]
    - sprawdza obsługi sygnały SIGTERM (Ciężarówka ma zakończyć prace)
    - symulacja komendy: `./truck 2 200 15`
- Test Truck 6 [link]
    - sprawdza obsługi sygnały SIGINT (Ciezarokwa ma przejść w tryb "CzyNagle")
    - symulacja komendy: `./truck 2 200 15`

Poprawki kodu po oddawaniu

Po analizie kodu okazało się, że funkcja która zajmuje się zarządzaniem ciężarówkami. istniała w tej samej pętli co odbieranie sygnału. Jeżeli sygnał nie był odbierany, to cegielnia nie sprawdzała czy może wysłać cegieł ciężarówką. Naprawiłem to tworząc dwa osobne wątki. Jeden wątek obsługuje odbieranie sygnałów, drugi obsługuje wyświetlanie stanu cegielni oraz zarządzanie wysyłaniem cegieł do ciężarówek. Poniżej znajduje się kod uruchamiania wątków oraz oczekiwania na ich zakończenie [link].

```
//rozpoczynanie pracy watkow
if (pthread_create(&thread_report, NULL, PetlaRaportuICiezarowek, NULL) !=
0) {
    perror("Failed to create thread");
    return 1;
}
if (pthread_create(&thread_worker_messages, NULL,
Watek_OdbieranieSygnalow, NULL) != 0) {
    perror("Failed to create thread");
    return 1;
}


//oczekiwanie na zakonczenie pracy watkow
if (pthread_join(thread_report, NULL) != 0) {
    perror("Nie udalo sie zakonczyc watkow");
    return 1;
}
if (pthread_join(thread_worker_messages, NULL) != 0) {
    perror("Nie udalo sie zakonczyc watkow");
    return 1;
}
```

Funkcja “Watek_OdbieranieSygnalow” [link] zajmuje się teraz jedynie odbieranie sygnałów.

```
void* Watek_OdbieranieSygnalow(void* arg){
    while (running) {
        // Otrzymywanie wiadomosci
        if (msgrcv(msgid, &msg, sizeof(msg) - sizeof(long), 0, 0) >= 0) {
            if(msg.messageType == MSG_WYSLANIE_CEGIEL){
                //msg.messageType = MSG_WYSLANIE_CEGIEL;
                //msg.senderType = SENDER_TYPE_WORKER;
                //msg.senderId = workerId;
                //msg.rozmiarCegly = 0;
                ceglyDodaj(msg.rozmiarCegly);
            }
        } else {
            perror("Blad otrzymywania wiadomosci");
            break;
        }
    }
    printf("Watek -> OdbieranieSygnalow zakończył pracę.\n");
    return NULL;
}
```

Funkcja “PetlaRaportuICiezarowek” [link] zajmuje się teraz jedynie wyświetlaniem stanu cegielni oraz zarządzaniem wysyłania cegieł ciężarówkami.

```
void* PetlaRaportuICiezarowek(void* arg){
    while (running) {
        ceglyRaport();
        wyslanieCiezarowek();
        sharedMemory->ceglyIlosc = ceglyIlosc;
        sharedMemory->ceglyWaga = ceglyWaga;
        sleep(1);
    }
    printf("Watek -> Stan cegielni i ciezarowki zakończył pracę.\n");
    return NULL;
}
```