Opis rozwiązania zadania Automat

Krzysztof Piesiewicz (kp385996)

1. Pliki rozwiązania

- `validator.c, tester.c, run.c`
- `err.c, err.h` – obsługa błędów na bazie bliblioteki err z labów
- `msg.c, msg.h` – obługa kolejek komunikatów
- `states.c, statec.h` – obłsuga stanów

2. Komunikacja validator – tester

Proces validator umieszcza dwie puste kolejki komunikatów – widma (/kp385996_validator_still_alive i /kp385996_tickets_available). Służą testerom do sprawdzenia czy serwer pracuje oraz czy prowadzi sprzedaż biletów. Każdy pobrany bilet z kolejki /kp385996_tickets_for_validation uprawdnia testera do wysłania jednego zlecenia do serwera. Serwer wydaje tylko jeden bilet aż do otrzymania kolejnego zgłoszenia (dzięki temu nie przyjmie więcej nowych zgłoszeń po otrzymaniu słowa '!'). Serwer przyjmuje zgłoszenia od testerów przez kolejkę /kp385996_validator_receiver.

Gdy serwer kończy wydawanie biletów, likwiduje kolejkę /kp385996_tickets_available oraz odblokowuje testerów z kolejki /kp385996_tickets_for_validation przez wysłanie odpowiedniej liczby komunikatów (niebędących biletami).

Przed zakończeniem pracy, serwer wysyła wszystkim testerom, którzy nie otrzymali od niego odpowiedzi na poszczególne zlecenia, wiadomości w liczbie brakujących komunikatów (testerzy zawsze oczekują na komplet wiadomości przed zakończeniem, chyba że nie istnieje już kolejka /kp385996_validator_still_alive).

3. Proces tester

Proces tester wczytuje z wejścia kolejne linie (również puste), zapisuje do buffora słów oraz przypisuje im numery – kolejne liczby całkowite od 0.

Zakładam, że wczytywanie wejścia jest nieblokujące, tj; nie wymagam od testera zakończenia pracy od razu po zakończeniu pracy serwera, gdy tester zawiesił się na operacja wejścia. Natomiat po każdym wczytaniu linii, tester sprawdza status serwera.

Program wypisuje słowa na wyjście w kolejności wczytania. Tester nie wypisuje słów, dla których nie otrzymał odpowiedzi lub dla których otrzymał informację o błędzie przetwarzania (błędy nie liczą się do liczby otrzymanych odpowiedzi).

4. Proces validator

Program wczytuje standardowe wejście w blokach i zapisuje je do poźniejszego rozsyłania procesom run.

Program nadaje numer każdemu komunikującemu się z nim testerowi (kolejne liczby całkowite od 0), co ułatwia przechowywanie informcji o zleceniach i statystykach, oraz przydaje się przy sprzątaniu po błędzie.
Dla każdego zlecenia od testera validator tworzy osobny proces run (gdy utworzenie procesu i łącz do komunikacji się nie powiedzie, serwer sprząta i kończy się z błędem – jednaka przedtem oczekuje na zakończenie wszystkich procesów run).

Proces validator wysyła wiadomości do testera przez kolejkę: /kp385996_res_for_tester_*[pid_testera]*.

5. Komunikacja validator – run

Proces validator za pomocą łącza nienazwanego przekazuje procesowi run na jego standardowe wejście: numer_testera, numer słowa oraz słowo, (ułatwia to późniejszą identyfikację wyniku otrzymanego przez validator).
Następnie w ten sam sposób przesyła opis automatu (wspomniane bloki wejścia).

Proces run przesyła validatorowi wynik poprzez kolejkę komunikatów: /kp385996_validator_receiver.

6. Proces run

Na stadardowe wejście wczytuje numer testera, numer słowa, słowo oraz opis automatu. Gdy słowo jest puste od razu zwraca wynik. W przeciwnym przypadku tworzy nieblokujące łącza nienazwane do komunikacji ze swoimi pracownikami (innymi prcesami run) oraz samych tych pracowników. Każdy pracownik odpowiada za gromadzenie podwyników dla jednego stanu.

Przy wystąpieniu błędu tworzenia łącz lub pracowników, proces run wysyła błąd do validatora i sprząta. Wystąpienia innych błędów nie zakładam. Aczkolwiek, gdyby implementacji automatu nie udało się obliczyć wyniku, to validator również dostanie błąd.

Pracownicy komunikują się ze sobą przez wspomniane łącza. Każdy pracownik posiada 3 łacza do odczytu (służące do odbierania: zleceń, wyników, innych komunikatów) oraz łącza do wysyłania wiadomości do wszystkich innych pracowników oraz do rodzica – procesu run.

Każdy pracownik w pętli odczutuje i wysyła wiadomości. Aby nie dochodziło do zawieszania procesów prowadzącego do zakleszczenia (przez ograniczony rozmiar łącz), każdy pracownik posiada własne stosy komunikatów do wysłania, z których próbuje wysyłać wiadomości.