Instrukcja do laboratorium Systemów Operacyjnych

(semestr drugi)

Ćwiczenie czwarte (dwa zajęcia)

Temat: Komunikacja IPC – kolejki komunikatów

Opracowanie:

mgr inż. Arkadiusz Chrobot

dr inż. Grzegorz Łukawski

1. Wprowadzenie

System Linux, podobnie jak inne systemy wywodzące się z Uniksa, udostępnia procesom użytkownika mechanizmy komunikacji, nazywane w skrócie IPC (*ang. Interprocess Communication*). W skład tych mechanizmów wchodzą kolejki komunikatów. Choć możliwe jest, że ten mechanizm zostanie zmieniony lub częściowo zastąpiony innym w przyszłych specyfikacjach standardu, to obecnie stanowi on dogodny sposób komunikacji między procesami. W systemach uniksowych istnieją dwa polecenia dostępne z poziomu powłoki użytkownika związane z obsługą komunikacji IPC. Pierwsze z nich to ipcs (szczegóły: man ipcs), które wyświetla informacje na temat utworzonych w systemie kolejek, semaforów i pamięci dzielonych. Drugie ipcrm (szczegóły: man ipcrm) służy do „ręcznego" usuwania z systemu zasobów IPC. W systemie Linux istnieje również zbiorcza dokumentacja poświęcona komunikacji IPC, wyświetlana poleceniem man 5 ipc.

2. Kolejki komunikatów

Kolejki komunikatów są tworzone w przestrzeni jądra systemu. Proces użytkownika ma do nich dostęp poprzez odpowiednie wywołania systemowe (dokładniej, poprzez odpowiednie funkcje biblioteczne języka C). Każda kolejka ma swój unikatowy identyfikator. Obsługa kolejek jest mniej skomplikowana niż łączy nazwanych. Ponadto sposób przesyłania informacji jest bardziej elastyczny. Jedna kolejka może służyć kilku procesom. Procesy mogą odbierać wszystkie komunikaty przechodzące przez kolejkę, lub tylko wybrane. Komunikaty podlegają ograniczeniom zarówno pod względem wielkości jednego komunikatu, jak i sumarycznej wielkości wszystkich komunikatów. W systemie Linux pierwszy limit wynosi 4KB (stała MSGMAX), a drugi 16KB (stała MSGMNB).

3. Pliki nagłówkowe związane z obsługą kolejek

Każdy program obsługujący kolejkę musi mieć włączone do swojego kodu źródłowego następujące plik nagłówkowe: *sys/types.h* – definicje typów, *sys/ipc.h* - funkcje i struktury wspólne dla wszystkich mechanizmów IPC, *sys/msg.h* – funkcje i struktury przeznaczone do obsługi kolejek komunikatów.

4. Najważniejsze struktury związane z kolejkami

Każdy komunikat opisany jest strukturą, w której znajduje się obowiązkowe pole określające typ komunikatu. Dalsza część definicji struktury komunikatu jest dowolna. Najczęściej podawana jest taka oto przykładowa struktura komunikatu:

```
struct msgbuf {
    long mtype;
    char mtext[1];
    };
```

Jądro systemu, dla każdej kolejki utrzymuje strukturę *msqid_ds*, którą można częściowo modyfikować za pomocą funkcji msgctl(), głównie dotyczy to pól struktury zagnieżdżonej *msg_perm*: *uid* – identyfikatora użytkownika dla właściciela, *guid* – identyfikator grupy dla właściciela oraz pola *mode*, które obejmuje 9 najmłodszych bitów określających tryb dostępu do kolejki.

Najważniejsze funkcje obsługujące kolejki komunikatów

- *ftok*() – funkcja zwraca unikatowy[1] identyfikator, który można użyć do tworzenia kolejki komunikatu. Aby dwa niespokrewnione procesy mogły korzystać z tego samego kanału komunikacyjnego muszą podać te same argumenty dla wywołania tej funkcji (tj. ścieżkę dostępu do dowolnie wybranego pliku i naturalną liczbę ośmiobitową[2]. Szczegóły: man ftok.
- *msgget*() – funkcja tworząca kolejkę komunikatów i zwracająca jej identyfikator lub zwraca identyfikator kolejki istniejącej. Kolejka jest tworzona na podstawie unikatowego klucza zwracanego przez funkcję ftok(), lub dobranego przez programistę. Jeśli kolejka ma być stworzona tylko i wyłącznie na użytek jednego procesu, to jako klucz podaje się stałą IPC_PRIVATE. Drugi argument wywołania tej funkcji określa tryb dostępu (czterocyfrowa liczba ósemkowa zaczynająca się od zera, lub odpowiednia kombinacja stałych MSG_R i MSG_W) oraz może określać w jaki sposób ma być uzyskany identyfikator kolejki (IPC_CREAT i IPC_EXCL). Szczegóły: man

1 Jest to założenie idealne, ale w praktyce zdarzają się kolizje. Oznacza to, ze dla dwóch różnych par argumentów wywołania funkcja ftok() może zwrócić takie same wartości.

2 Chociaż typem argumentu jest long, to branych jest pod uwagę tylko osiem najmłodszych bitów wartości argumentu.

msgget

- *msgsnd*() – funkcja umożliwia dodanie komunikatu do kolejki. W zależności od wartości ostatniego argumentu może, w przypadku kiedy kolejka jest zapełniona, oczekiwać na zwolnienie miejsca lub zwracać błąd (IPC_NOWAIT). Szczegóły: man msgsnd
- *msgrcv*() – funkcja umożliwia odbiór komunikatu z kolejki. Możliwe jest selektywne odbieranie komunikatów, w zależności od wartości argumentu funkcji określającego typ odbieranego komunikatu. Jeśli będzie on miał wartość zero, to będzie odebrany pierwszy komunikat znajdujący się w kolejce. Jeśli wartość dodatnią, to odebrany będzie pierwszy komunikat o takim samym typie. Jeśli natomiast argument będzie miał wartość ujemną, to odebrany będzie komunikat o typie takim samym lub mniejszym co do wartości bezwzględnej od podanego argumentu. Szczegóły: man msgrcv.
- *msgctl*() – funkcja umożliwiająca sterowanie istniejącą kolejką komunikatów. Może ona wykonać trzy operacje: IPC_STAT – pobranie do struktury msqid_ds informacji o kolejce, IPC_SET – ustawienie danych kolejki na podstawie zawartości struktury msqid_ds i IPC_ RMID – usunięcie kolejki. Ostatnia operacja jest możliwa tylko wtedy, kiedy procesy korzystające z kolejki zakończyły operacje pisania i czytania. Szczegóły: man msgctl.

Zadania

1. Napisz program, który utworzy prywatną kolejkę i będzie przesyłał nią komunikaty.
2. Zmodyfikuj zadanie pierwsze, tak aby proces tworzył potomka i komunikował się z nim przy użyciu kolejki.
3. Napisz dwa programy (procesy niespokerwnione), które będą komunikować się przy pomocy kolejki komunikatów. Do wygenerowania klucza użyj funkcji *ftok()*.
4. Zmodyfikuj programy z zadania trzeciego tak, aby zbadać jak się będzie zachowywała funkcja *msgrcv()*, w zależności od tego, czy otrzyma flagę IPC_NOWAIT, czy nie.
5. Napisz dwa programy: pierwszy wyśle kilka komunikatów o losowo wy-

branym typie (przyjmijmy, że typy komunikatów należą do przedziału [1,5]), a drugi będzie je odbierze w porządku malejącym lub rosnącym, ze względu na wartość typu.

6. Napisz dwa programy: pierwszy wyśle dziesięć komunikatów, a drugi w zależności od parametru wywołania odbierze je w takiej samej lub odwrotnej kolejności do tej z jaką wysłał je pierwszy program.
7. Napisz trzy programy, które komunikowałyby się przez wspólną kolejkę. Komunikacja musi być dwukierunkowa. Każdy program musi odbierać informacje przeznaczone tylko dla niego, ale może wysyłać informacje do obu pozostałych programów.
8. Za pomocą kolejki komunikatów programy mogą przesyłać wiadomości o różnej długości. Napisz dwa programy komunikujące się przez jedną kolejkę, ale przesyłające między sobą komunikaty o zmiennej długości. Załóż, że każdy „właściwy" komunikat będzie poprzedzany komunikatem o ustalonym formacie, który będzie informował o wielkości następującego po nim komunikatu.
9. Napisz trzy programy. Pierwszy niech zapisuje do kolejki liczby parzyste, drugi nieparzyste, a trzeci niech odczytuje kolejne pary liczb z kolejki i niech je sumuje.

Uwaga: We wszystkich programach tuż przed zakończeniem ich działania wszystkie kolejki komunikatów z jakich one korzystają powinny zostać usunięte.