SCZR lab. 2

# Realizacja systemu wykorzystującego komunikację międzyprocesową w czasie rzeczywistym w systemie Linux

Celem tego ćwiczenia jest doświadczalne zbadanie działania mechanizmów komunikacji międzyprocesowej w systemie Linux, ze szczególnym uwzględnieniem opóźnień jakie występują w komunikacji między współpracującymi procesami.

## Materiał do przejrzenia

Slajdy z wykładu 3, w razie konieczności także przypomnienie materiałów z wykładów 1 i 2.
Źródła programów **cw2a** i **cw2b**, wykorzystywanych w ćwiczeniu (udostępnione na serwerze studia w archiwum *sczr_lab2_student.tar.bz2*).
Oczywiście może być konieczne samodzielne uzupełnienie wiadomości niezbędnych do realizacji ćwiczenia.

## Wprowadzenie

Jedną z typowych sytuacji, w których występuje konieczność komunikacji w czasie rzeczywistym między procesami, jest akwizycja i przetwarzanie danych. Sprzęt pomiarowy może być obsługiwany przez jeden proces, który odbiera zmierzone dane, a następnie udostępnia je jako serwer programom-klientom, które będą je przetwarzać. W takim scenariuszu istotny jest czas, jaki upływa między pozyskaniem danych przez serwer, a rozpoczęciem ich przetwarzania przez klientów. W tym ćwiczeniu można zaobserwować jak ten czas zależy od liczby rdzeni CPU, liczby programów klienckich, stopnia obciążenia systemu (wynikającego z czasu przetwarzania danych) oraz sposobu oczekiwania klienta na dostępność danych.

## Przebieg ćwiczenia

Dany jest zestaw programów, symulujący pracę prostego systemu zbierającego i przetwarzającego dane.
Program **cw2a** powinien być uruchamiany następująco:

```
cw2a liczba_klientów liczba_próbek okres_próbkowania czas_przetwarzania
```

Okres próbkowania podawany jest w mikrosekundach, a czas przetwarzania w jednostkach umownych (jest to liczba wykonań pętli).
Sugerowana wartość okresu próbkowania to 10000 (czyli 100 zestawów próbek na sekundę). Skrócenie tego okresu może pozwolić na szybsze przeprowadzenie pomiarów, jednak ewentualna zbyt duża częstotliwość przełączania procesów może dodatkowo zakłócić wyniki.

Program **cw2a** uruchamia zadaną liczbę programów przetwarzających dane **cw2b**, a następnie rozpoczyna generację symulowanych “zestawów próbek” i przekazuje je przez pamięć dzieloną do programów **cw2b**. Po wygenerowaniu i odebraniu zleconej liczby zestawów próbek cały system kończy pracę.
Program **cw2a** generuje zbiór `server.txt`, w którym zapisane są podane w mikrosekundach momenty “pobrania” kolejnych zestawów próbek (1.01.1970, godz. 0:00 odpowiada wartości 0).
Programy **cw2b** generują zbiory `cli_N_DELAY_NCPU.txt` (N – numer klienta, DELAY - wartość parametru delay, NCPU - liczba rdzeni), w których zapisane są momenty pobrania i dostarczenia kolejnych zestawów próbek oraz różnica tych czasów (czyli opóźnienie dostarczenia danych).

**Uwaga! Klient cw2b musi być dostępny w ścieżce PATH. W przypadku OpenWRT będzie on instalowany w /usr/bin, ale przy testach na “gospodarzu” może być konieczne dodanie właściwego katalogu do ścieżki PATH.**

**Uwaga! Podczas tego ćwiczenia często będzie potrzebne przenoszenie plików między maszyną wirtualną i "gospodarzem". Dlatego proszę uruchomić QEMU z wykorzystaniem systemu plików PLAN9 (9p). Programy testowe można uruchamiać w zamontowanym katalogu, dzięki czemu wyniki będą od razu dostępne dla "gospodarza".**

**Wskazówka: Do analizy plików z wynikami i sporządzenia wykresów można użyć np. programu oocalc (localc) z pakietu OpenOffice (LibreOffice), dostępnej przez sieć aplikacji Google Spreadsheet, albo programu Octave.**

**Plan badań**

1. Proszę przetestować działanie programów na "gospodarzu"
2. Proszę skompilować pakiet przy pomocy SDK OpenWRT i zainstalować go w systemie plików maszyny wirtualnej.
3. Proszę zaobserwować, dla jakiej wartości czasu przetwarzania danych, system przestaje nadążać przetwarzać dane w czasie rzeczywistym (będzie to widoczne jako ciągły wzrost czasu opóźnienia). Testy proszę przeprowadzić w różnych kombinacjach dla 1 i 3 klientów, dla maszyny wirtualnej mającej 1, 2 lub 4 rdzenie CPU (*można to modyfikować przez argument -smp przy wywołaniu QEMU)*.
4. Dla czasu przetwarzania danych, stanowiącego połowę wartości, przy której system przestaje nadążać z przetwarzaniem, proszę zbadać rozkład czasu dostarczenia danych do klienta (dla tych samych kombinacji liczby klientów i rdzeni CPU co w punkcie 3). Wyniki proszę przedstawić w formie histogramów.
5. Proszę zmodyfikować aplikację kliencką **cw2b** tak, aby zamiast oczekiwać w uśpieniu na dane, oczekiwała aktywnie. Proszę przygotować dwa warianty modyfikacji. W jednym zmiana powinna dotyczyć tylko klienta numer 0, w drugim wszystkich klientów. Proszę zaobserwować, jak to wpłynie na obserwowany rozkład czasu dostarczenia danych.
6. Analizując wygenerowany zbiór "server.txt" proszę sprawdzić, czy rzeczywiście okres między pobraniami zestawów próbek jest właściwy (proszę zbadać jego rozkład). Proszę wyjaśnić obserwowany efekt. Proszę zmodyfikować aplikację **cw2a** tak, aby go wyeliminować i przedstawić uzyskane wyniki.

**Cennik**

zadanie 1 – 0 do 1 pkt
zadanie 2 – 0 do 2 pkt
zadanie 3 – 0 do 3 pkt
zadanie 4 – 0 do 3 pkt
zadanie 5 – 0 do 3 pkt
zadanie 6 – 0 do 3 pkt
czytelność kodu, komentarze – -5 do 0 pkt
prezentacja wyników – -10 do 0 pkt

Maximum: 15 pkt.