Systemy komputerowe w sterowaniu i pomiarach (SKPS)
Instrukcja do laboratorium 4

dr hab. inż. Wojciech Zabołotny, mgr inż. Dawid Seredyński
Ostatnia aktualizacja: 21.04.2022

[Informacje wstępne](#)
- [Cel laboratorium](#)
- [Zakres wejściówki](#)
- [Plan laboratorium 4:](#)
- [Praca na zajęciach](#)
- [Oznaczenia użyte w tym dokumencie](#)
- [Materiały](#)

[Wprowadzenie](#)

[Opis programów używanych podczas laboratorium](#)

[Zadania](#)
- [1. Przetestowanie działania programów na “gospodarzu”.](#)
- [2. Zbudowanie pakietu dla OpenWRT](#)
- [3. Ustalenie granicznej wartości czasu przetwarzania](#)
- [4. Rozkład czasu dostarczenia danych](#)
- [5. Aktywne oczekiwanie](#)
- [6. Właściwy pomiar czasu](#)

Informacje wstępne

Cel laboratorium

Realizacja systemu wykorzystującego komunikację międzyprocesową w czasie rzeczywistym na przykładzie OpenWRT i Raspberry Pi.

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

Laboratorium 4 jest za 7 punktów, w tym:
- 1 - wejściówka (UWAGA: osoby, które otrzymają 0 punktów z wejściówki mogą nie zostać dopuszczone do dalszej części laboratorium)
- 6 - praca na zajęciach

Zakres wejściówki

1. Materiały z wykładu 4: komunikacja międzyprocesowa, uruchamianie procesów, synchronizacja dostępu, priorytety procesów
2. Histogram - co to jest

Plan laboratorium 4:

1. Pobranie i zbudowanie pakietu OpenWRT do laboratorium 4.
2. Przeprowadzenie badań zgodnie z instrukcją

Praca na zajęciach

W trakcie zajęć proszę na bieżąco tworzyć raport z wykonanych prac. Kolejne etapy należy przedstawiać prowadzącemu.

UWAGA: tym razem, wyjątkowo, raport z laboratorium 4 musi zostać zakończony podczas zajęć. Jest to spowodowane koniecznością zebrania, opracowania i analizy danych, które można uzyskać tylko ze sprzętem dostępnym na laboratorium.

Oznaczenia użyte w tym dokumencie

Polecenia, które są wpisywane w konsoli **Linuxa** na komputerze **host (PC)** oznaczone są jasnoszarym tłem, np.:
`ls -la /`

Polecenia, które są wpisywane w konsoli **Linuxa** na komputerze **RPi** oznaczone są malinowym tłem (raspberry, czyli malina), np.:
`ls -la /`

Materiały

1. Slajdy z wykładu 4
2. Pakiet OpenWRT z programami cw4a i cw4b (udostępnione na Moodle w sekcji *Pliki do laboratorium*, plik *skps_lab4_student.tar.xz*): https://moodle.usos.pw.edu.pl/pluginfile.php/241562/mod_folder/content/0/skps_lab4_student.tar.xz?forcedownload=1

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.

Opis programów używanych podczas laboratorium

Dany jest zestaw programów (w ramach pakietu OpenWRT, do samodzielnego zbudowania), symulujący pracę prostego systemu zbierającego i przetwarzającego dane.

Program `cw4a` powinien być uruchamiany następująco:
`cw4a 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 `cw4a` uruchamia zadaną liczbę programów przetwarzających dane (program `cw4b`), a następnie rozpoczyna generację symulowanych “zestawów próbek” i przekazuje je przez pamięć dzieloną do uruchomionych instancji `cw4b`. Po wygenerowaniu i odebraniu zleconej liczby zestawów próbek cały system kończy pracę.

Program `cw4a` generuje zbiór *server.txt*, w katalogu roboczym, w którym zapisane są podane w mikrosekundach momenty “pobrania” kolejnych zestawów próbek (1.01.1970, godz. 0:00 odpowiada wartości 0).
Programy `cw4b` generują zbiory *cli_N.txt* (N – numer klienta), w których zapisane są momenty pobrania i dostarczenia kolejnych zestawów próbek oraz różnica tych czasów (czyli opóźnienie dostarczenia danych).

Uwaga: Klient `cw4b` musi być dostępny w zmiennej środowiskowej *PATH*. W przypadku OpenWRT, programy zostaną zainstalowane w /usr/bin (ścieżka ta jest w *PATH*), ale przy testach na “gospodarzu” może być konieczne dodanie właściwej ścieżki do zmiennej *PATH*. Do analizy plików z wynikami i sporządzenia wykresów można użyć np. programu Calc z pakietu LibreOffice, dostępnej przez sieć aplikacji Google Spreadsheet, albo programu Octave.

Zadania

1. Przetestowanie działania programów na “gospodarzu”.

Programy można zbudować poleceniem `make` wywołanym w folderze z plikami źródłowymi. Przed uruchomieniem należy pamiętać o ustawieniu właściwej ścieżki w zmiennej PATH.

2. Zbudowanie pakietu dla OpenWRT

Proszę skompilować pakiet przy pomocy SDK OpenWRT i zainstalować go na RPi.

UWAGA: Przed zbudowaniem pakietu w OpenWRT SDK należy usunąć wyniki poprzedniego budowania (na “gospodarzu”) poleceniem `make clean`, w folderze ze źródłami.

3. Ustalenie granicznej wartości czasu przetwarzania

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 dla następujących wariantów:

1. Pierwszy wariant: 3 klientów, 1 rdzeń, pełne obciążenie
2. Drugi wariant: 3 klientów, 2 rdzenie, pełne obciążenie
3. Trzeci wariant: 3 klientów, 2 rdzenie, bez obciążenia
4. Czwarty wariant: 1 klient, 4 rdzenie, bez obciążenia

Pełne obciążenie można uzyskać np. wykonując w tle proces (pełne obciążenie dla wszystkich rdzeni przez 1 minutę - można też ustawić dłuższy czas):
`stress-ng --matrix 0 -t 1m`

Liczbę dostępnych rdzeni dla RPi można ustawić w pliku *cmdline.txt* na partycji boot w katalogu *user* dodając parametr `maxcpus=N`, gdzie `N` to liczba dostępnych rdzeni. W przypadku qemu liczbę dostępnych rdzeni można zmienić za pomocą argumentu `-smp` przy wywołaniu qemu.

4. Rozkład czasu dostarczenia danych

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 zadaniu 3. Wyniki proszę przedstawić w formie histogramów.

5. Aktywne oczekiwanie

Proszę zmodyfikować aplikację kliencką `cw4b` 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. Właściwy pomiar czasu

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 `cw4a` tak, aby wyeliminować obserwowany efekt i przedstawić uzyskane wyniki.