Katedra
Informatyki i Automatyki
Politechniki Rzeszowskiej

# Informatyka

## 10. Systemy Operacyjne

Opracował: Maciej Penar

# Spis treści

# 1. Zagadnienia teoretyczne

1. Wyjaśnij relację pomiędzy: **procesem** a **wątkiem**
2. Wyjaśnij różnicę pomiędzy: **współbieżnością kooperacyjną** a **współbieżnością konkurencyjną**
3. Wyjaśnij różnicę (o ile jest) pomiędzy **współbieżnością** a **równoległością**
4. Co to jest fork-bomba?
5. Co to jest atak typu DoS? Co to jest atak typu DDoS?
6. Do czego służą programy w systemach unixopodobnych:
    a. ps
    b. kill
    c. pstree
    d. top
7. Co to jest mutex? (patrz pkt 4).

# 2. Przygotowanie do zajęć (pkt. 3)

Na potrzeby 3) wybierz jedno z poleceń:

- Zainstaluj / uzyskaj dostęp do unix'a lub systemu unixo'podobnego wraz z możliwością kompilacji C/C++
- Wejdź na stronę: https://www.onlinegdb.com/online_c++_compiler (tu semafory nie zadziałają)
- Wejdź na stronę: https://ideone.com (a tu zadziałają)

# 3. Procesy: Fork()

Ta część wymaga 2) i zadziała tylko na unixie.

1. Co robi poniższy kod?

```
#include <iostream>
#include <unistd.h>

int main() {
        pid_t pid;
        pid = fork();
        if (pid == -1) {
                std::cout << "Exit with failure" << std::endl;
                exit(EXIT_FAILURE);
        }

        if (pid == 0) {
                std::cout << "Child!" << std::endl;
        }
        else {
                std::cout << "Parent!" << std::endl;
        }
        sleep(3);
        return 0;
}
```

2. Co robi poniższy kod? Ile komunikatów łącznie się pojawi? Ile procesów w sumie weźmie udział w przetwarzaniu?

```
#include <iostream>
#include <unistd.h>

void shout(int id, bool child) {
        std::cout << "I am process no. " << id << (child ? " and I'm a child" :
"") << std::endl;
}

void aux(int n) {
        if (n > 0) {
                pid_t pid;
                pid = fork();
                if (pid == -1) {
                        std::cout << "Exit with failure" << std::endl;
                        exit(EXIT_FAILURE);
                }

                shout(pid, pid == 0);
                aux(n - 1);
        }
}

int main() {
        aux(2);
        sleep(1000);
        return 0;
}
```

3. Na bazie programów 1) oraz 2) napisz program który uruchomi trzy procesy (w tym miejscu patrzę na zadanie 4.2 i wiem że trzeba zdekomponować każdy podpunkt na osobną funkcję):
    a. W ramach pierwszego nastąpi wypisanie pierwszych 5 wielokrotności liczby 2.
    b. W ramach drugiego nastąpi wypisanie 7 razy „...ups”
    c. W ramach trzeciego nastąpi wypisanie: „Jestem X kebabem”. Za X podstawić w kolejnych iteracjach (począwszy od iteracji zerowej): „mały”, „średnim”, „dużym”.

4. Przeanalizuj poniższy kod (dzięki uprzejmości dr Sławomira Samoleja: http://ssamolej.kia.prz.edu.pl/dydaktyka/inf_1EE_ZI/Procesy_wspolbieznosc1.zip):

```
#include <iostream>
#include <unistd.h>
#include <sys/file.h>
#include <semaphore.h>

int main()
{
        int a = 0;
        pid_t pid;
        sem_t * sem = sem_open("my_semaphore", O_CREAT);

        sem_init(sem, 1, 0);
        pid = fork();

        if (pid == -1)
        {
                exit(EXIT_FAILURE);
        }

        if (pid == 0)
        {
                while (1)
                {
                        sem_wait(sem);
                        std::cout << "Synchronizing" << std::endl;
                }
        }
        else
        {
                for (int i = 0; i < 5; i++)
                {
                        std::cout
                                << "Notifying the semaphore "
                                << i
                                << std::endl;
                        sem_post(sem);
                        sleep(1);
                }
        }
        exit(EXIT_SUCCESS);
}
```

# 4. Wątki: #thread

1. Przeanalizuj ten fragment (powinien zadziałać na każdym systemie operacyjnym).

```
#include <iostream>
#include <mutex>
#include <thread>
#include <chrono>

std::mutex mtx;

void f(int iterations) {
        for (int i = 0; i < iterations; ++i) {
                mtx.lock();
                std::cout << i << std::endl;
                mtx.unlock();
                std::this_thread::sleep_for(std::chrono::seconds(1));
        }
}

int main(){
        const int THREADS = 7;
        std::thread temp[THREADS];
        for (int i = 0; i < THREADS; ++i) {
                temp[i] = std::thread(f, 10); //Przekazanie funkcji i parametru
        }
        std::this_thread::sleep_for(std::chrono::seconds(60));
        return 0;
}
```

2. Powtórz zadanie 3.3 (to z Kebabami). Ale zamiast procesów chodzi o wątki.
3. Zapoznać się z problemem stołujących filozofów. Znaleźć twist fabularny w zadaniu. Na repozytorium jest kod w pliku philosophers.cpp. Kilka podpowiedzi:
    a. Jak umiecie – to poprawcie kod tak żeby wszyscy się najedli.
        i. W main() – 1 linijka
        ii. Pozostały kod: ok. 12-14 linijek copy-paste
    b. W kodzie nie powinno być zbędnych słów kluczowych

    Link: https://github.com/mpenarprz/InformatykaA1/blob/master/kod/philosophers.cpp