**Obraz zawierający tekst, Czcionka, logo, wizytówka

Opis wygenerowany automatycznie**

**Systemy operacyjne 2**

*Mikroreferat: Obsługa procesów defunct przez init*

Prowadzący: Autorzy:

dr Krzysztof Balicki Kamil Krukar

Kierunek: Informatyka, grupa Lab1

Rzeszów, r.a. 2023/2024

W momencie gdy rodzicem procesu zostaje init, zostaje na nim natychmiast wykonana funkcja wait przez init. Ilustrują to poniższe programy:

**Program tworzący proces zombie (create\_zombie.c):**

Program ten rozpoczyna się od wypisania informacji o procesie macierzystym, który tworzy proces potomny. Wykorzystując funkcję fork, tworzy nowy proces potomny, który wykonuje krótką operację i następnie kończy swoje działanie za pomocą funkcji exit. Istotne jest, że proces macierzysty nie oczekuje na zakończenie procesu potomnego, co skutkuje tym, że proces potomny staje się procesem zombie. Program ten demonstruje, że nieobsłużony proces potomny pozostaje w stanie zombie po zakończeniu swojego działania.

**Program wyświetlający informacje o procesach zombie (show\_zombies.c):**

Drugi program skupia się na monitorowaniu procesów zombie. Po chwilowym oczekiwaniu na systemowe zaktualizowanie statusu procesów (za pomocą funkcji sleep), program wywołuje polecenie systemowe ps aux | grep Z, aby wyświetlić informacje o procesach oznaczonych jako zombie. Dzięki temu zobaczymy, jak procesy zombie są identyfikowane i śledzone przez system operacyjny.

Podsumowując, te dwa programy w połączeniu rzucają światło na procesy potomne, które stają się zombie i ilustrują, jak można monitorować te procesy za pomocą systemowych poleceń. Ręczne zarządzanie procesami potomnymi, takie jak obsługa procesów zombie, jest kluczowe w utrzymaniu stabilności systemu operacyjnego. W rzeczywistych scenariuszach te mechanizmy są obsługiwane przez system operacyjny, w tym przez proces init, aby zapewnić sprawne i stabilne funkcjonowanie systemu.

**Kod programu make\_zombies.c:**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

printf("Początek programu:\n");

system("ps aux");

printf("\n");

sleep(1);

pid\_t child\_pid = fork();

if (child\_pid == -1) {

perror("fork");

exit(EXIT\_FAILURE);

}

if (child\_pid == 0) {

// Kod procesu potomnego

printf("Proces potomny (PID=%d) działa.\n\n", getpid());

exit(EXIT\_SUCCESS); // Proces potomny kończy działanie

} else {

printf("Proces macierzysty (PID=%d) zakończył działanie, proces potomny (PID=%d) jest teraz zombie.\n", getpid(), child\_pid);

sleep(2); // Czekamy, aby dać czas systemowi na zaktualizowanie statusu

// Informacja o procesie zombie przed obsługą przez init

printf("\nPrzed obsługą przez init:\n");

system("ps aux | grep Z");

}

return 0;

}

**Kod programu show\_zombies.c:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

printf("Początek programu wyświetlającego procesy zombie:\n");

sleep(2); // Czekamy, aby dać czas systemowi na zaktualizowanie statusu

system("ps aux | grep Z");

return 0;

}

**Sposób kompilacji:**

gcc create\_zombie.c -o create\_zombie

gcc show\_zombies.c -o show\_zombies

./create\_zombie

./show\_zombies