|  |  |
| --- | --- |
| **Politechnika Świętokrzyska w Kielcach**  **Wydział Elektrotechniki, Automatyki i Informatyki** | |
| Laboratorium Programowanie Współbieżne | |
| Temat:  **Poznajemy funkcje fork, wait i signal** | Autor: **Rafał Robak**  Grupa: **3ID12A** |
| Numer laboratorium: **2** | Data wykonania : **13.11.2020** |

1. **Ściągawka**

**fork()** - służy do stworzenia nowego procesu w systemie Linux. W wyniku dostaje dwie wartości – proces potomny dostaje 0, proces macierzysty dostaje identyfikator dziecka, jeśli się nie uda wywołać fork() to zwraca on -1.

**wait() -** służy do oczekiwania na zakończenie potomstwa i odebranie jego statusu

**signal()** - funkcja służy do przechwycenia sygnału

**kill()** - służy do wysyłania sygnałów

1. Napisać/przepisać i uruchomić poniższy program, zrozumieć jak działa.

#include <stdlib.h>

#include <stdio.h>

main()

{

int childpid;

int status=1;

int zakonczono;

printf("startuje proces macierzysty pid %d\n",getpid());

if ((childpid = fork()) == -1)

{

perror("nie moge forknac");

exit(1);

}

else

if (childpid ==0 )

{

printf("Proces potomny o pidzie %d z rodzica %d\n",getpid(),getppid());

}

else

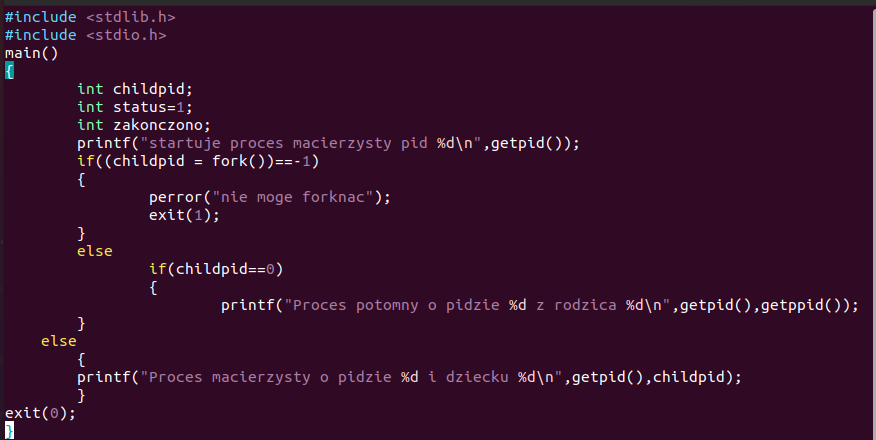
{

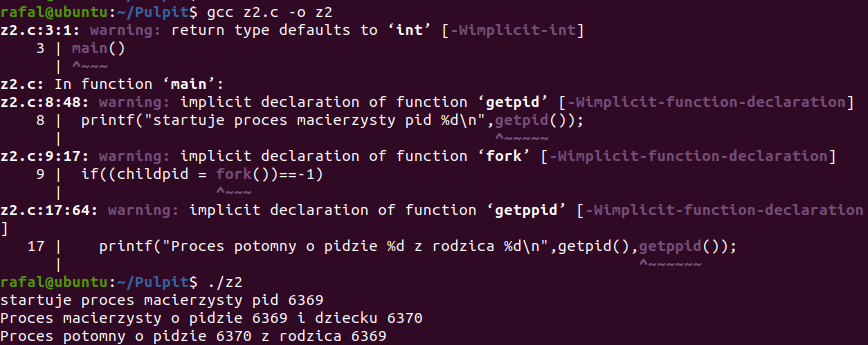
printf("Proces macierzysty o pidzie %d i dziecku %d\n",getpid(),childpid);

}

exit(0);

}

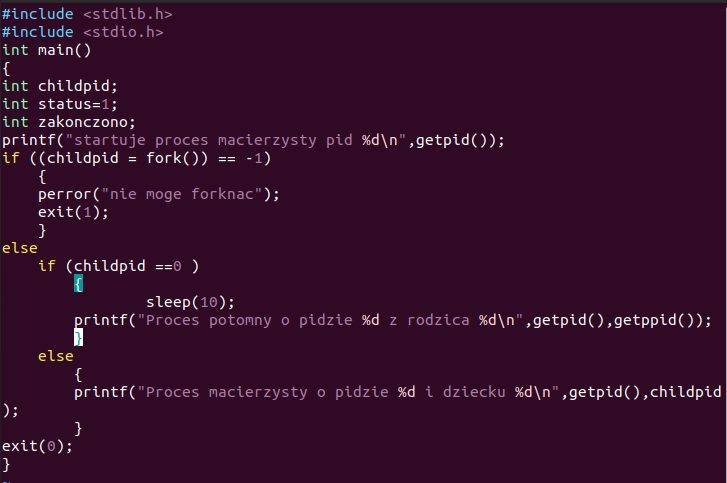


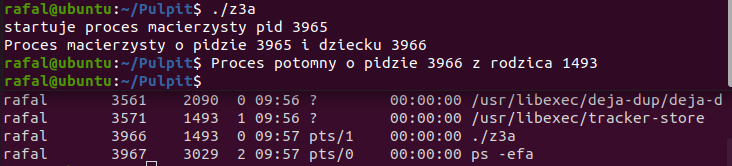


1. **Sprawić by proces potomny zakończył się po procesie macierzystym [0,5pkt] (funkcja sleep).**

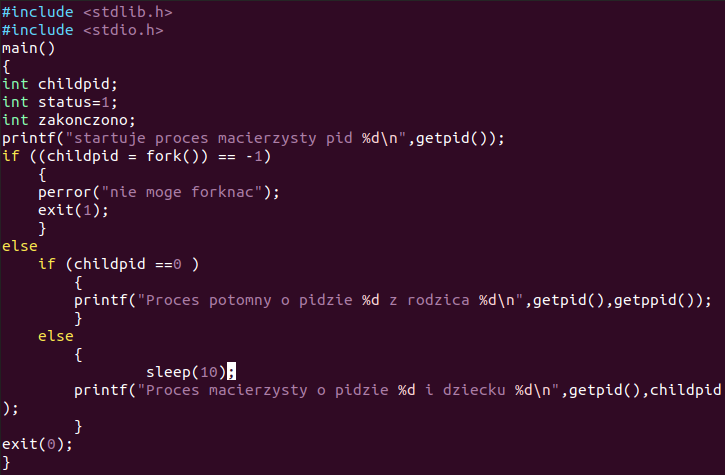
Sprawdzić identyfikator procesu macierzystego w procesie potomnym po zakończeniu rodzica. Sprawić by proces potomny zakończył się przed końcem rodzica (rodzic jeszcze nie wywołuje wait). Zbadać czy proces potomny widnieje jeszcze w systemie (ps -efa).

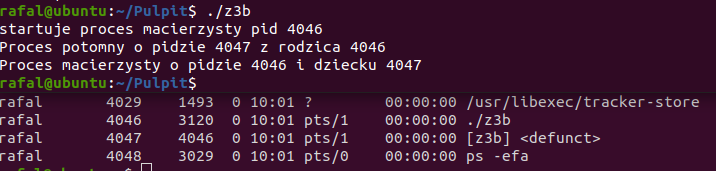
-Sleep w procesie potomnym

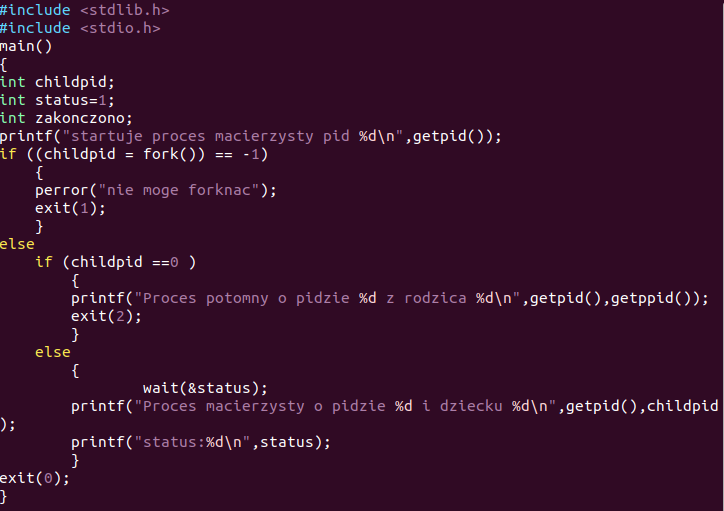


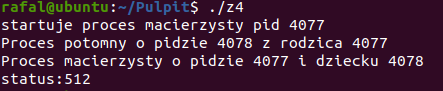


-Sleep w procesie macierzystym



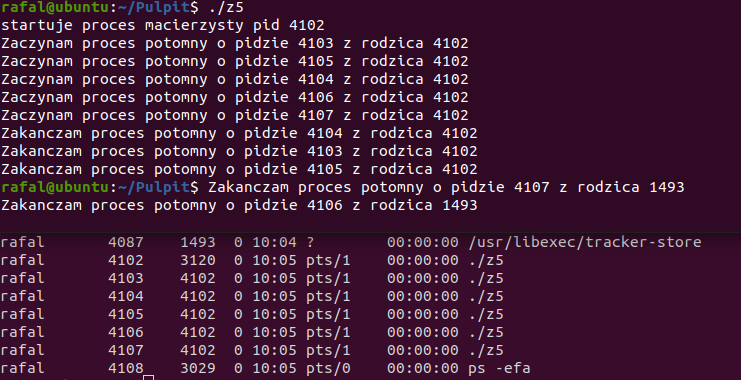


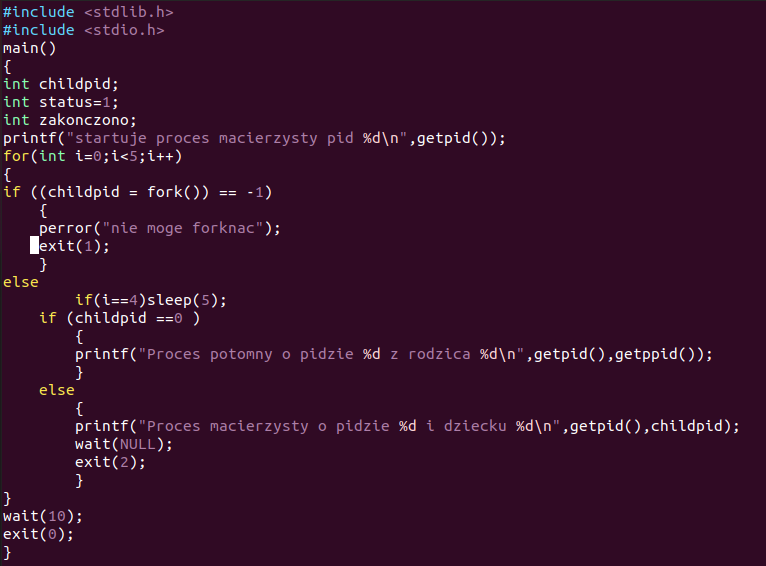
1. **Zastosować funkcję *wait*** [0,5pkt]  
   odbierając wynik zakończenia procesu potomnego. Sprawdzić zachowanie się obu procesów. Proces potomny powinien zwrócić wartość inną niż 0 np. exit(2). Wyświetlić co odbierze wait w sposób **niesformatowany** (nie używać  WEXITSTATUS czy przesunięć bitowych).  
   

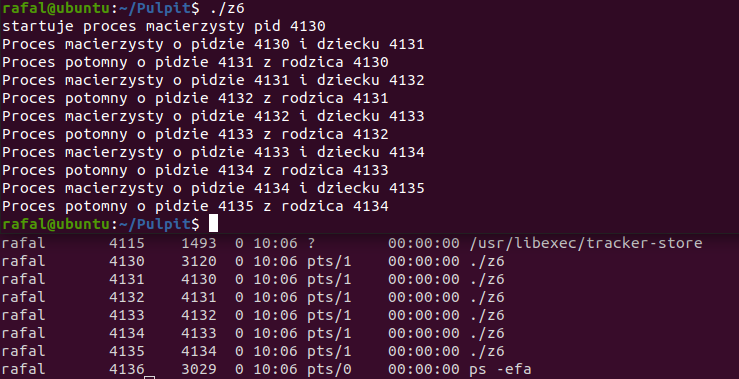


1. **Stworzyć n=5 procesów potomnych z jednego procesu macierzystego.** [1,5pkt]  
   Zastosować funkcję *wait* do poprawnego zakończenia. Każdy potomek powinien zatrzymać się na 5 sekund (*sleep*) by na drugiej konsoli sprawdzić za pomocą ps -efa czy udało się wygenerować dokładnie 5 potomków (ps powinien pokazać 5 procesów potomnych). Procesy powinny zaczynać się razem i kończyć razem. Zastosować pętlę zamiast konstrukcji if else if else ...





1. **Stworzyć n=5 pokoleń** [1,5pkt]  
   rodzic->potomek1->potomek2->...->potmek5. Uwaga występuje ryzyko zapętlenia tworzenia procesów. Zadbać o to by nie stracić kontroli nad systemem. Zapisać też edytowane pliki. Sprawdzić jak powyżej, czy udało się wygenerować dokładnie 5 potomków (ps powinien pokazać 5 procesów potomnych). Procesy powinny zaczynać się razem i kończyć razem. Zastosować pętlę zamiast konstrukcji if else if else ...  
   



1. **Przykład**[1pkt]  
   przechwycenia sygnału *SIGCHLD* - wysyłanego przez dziecko w momencie zakończenia się. Zastosować poniższy przykład w dowolnym z wykonanych programów.

