Tworzenie i kończenie procesów

To laboratorium ma na celu zapoznanie się z funkcją fork, wait oraz na sposobie przechwytywania sygnałów (funkcja signal)

1.Wstęp

Jedynym sposobem na stworzenie nowego procesu w systemie Unix jest wywołanie funkcji systemowej fork(). (man fork) inne funkcje np. clone vfork są interfejsami do fork'a. Funkcją związaną ściśle z forkiem jest wait czekanie na zakończenie potomstwa.
Sygnały. Do przechwycenia sygnału służy funkcja signal. Do wysyłania sygnałów służy funkcja kill.

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

```
#include <stdio.h>
#include <stdlib.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);
}
```

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

4.Zastosować funkcję wait 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.

5.Stworzyć n=5 procesów potomnych z jednego procesu macierzystego. Zastosować funkcję wait do poprawnego zakończenia. Każdy potomek powinien zatrzymać się na "chwilę" (sleep). Sprawdzić czy udało się wygenerować dokładnie 5 potomków (ps powinien pokazać 5 procesów potomnych). Zastosować pętlę zamiast konstrukcji if else if else ...

6.Stworzyć n=5 pokoleń rodzic->potomek1->potomek2->...->potmek5. Uwaga występuje ryzyko zapętlenia tworzenia procesów. Zadbać o to by nie stracić kontroli nad systemem. Zapisać też edytowane pliki. Sprawdzić czy udało się wygenerować dokładnie 5 potomków (ps powinien pokazać 5 procesów potomnych). Zastosować pętlę zamiast konstrukcji if else if else ...

7.Przykład przechwycenia sygnału SIGCHLD - wysyłanego przez dziecko w momencie zakończenia się. Zastosować poniższy przykład w dowolnym z wykonanych programów.

```
#include  <signal.h>/* w tym pliku jest definicja sygnalow */
/* przykładowa funkcja obsługi sygnału */
void oblsluga_zakonczenia_dziecka(int nr_sig)
{
printf("Rodzic już się dowiedział o zakończeniu procesu %d\n",wait(NULL));
}
/* poinformowanie procesu by w razie nadejścia sygnału SIGCHLD wywołał funkcję
obsluga_zakonczenia_dziecka */
signal(SIGCHLD,oblsluga_zakonczenia_dziecka);
```

8.Sprawozdanie

Nie zapomnijcie o sprawozdaniu