

Rodzaje sygnałów: SIGINT, SIGQUIT, SIGKILL, SIGTSTP, SIGSTOP, SIGTERM, SIGSEGV, SIGHUP, SIGALARM, SIGCHLD, SIGUSR1, SIGUSR2

Sygnały czasu rzeczywistego: SIGRTMIN, SIGRTMIN+n, SIGRTMAX

Przydatne polecenia Unix: kill, ps

Przydatne funkcje systemowe: kill, raise, sigqueue, signal, sigaction, sigemptyset, sigfillset, sigaddset, sigdelset, sigismember, sigprocmask, sigpending, pause, sigsuspend

Zadanie 1 (30%)

Napisz program demonstrujący, czy ustawienia dyspozycji dla sygnałów, ich maski oraz czekające sygnały są dziedziczone po wykonaniu funkcji *fork* oraz *exec*.

W szczególności eksperymenty proszę wykonać dla sygnału *SIGUSR1* w następujący sposób:

- *Dziedziczenie ustawień sygnałów po wykonaniu funkcji fork.* Proszę napisać program, który w zależności od wartości argumentu z linii poleceń, który może przyjmować wartości *ignore*, *handler*, *mask* lub *pending*, odpowiednio w procesie przodka ustawia ignorowanie, instaluje handler obsługujący sygnał wypisujący komunikat o jego otrzymaniu, maskuje ten sygnał oraz sprawdza (przy zamaskowaniu tego sygnału) czy wiszący/oczekujący sygnał jest widoczny w procesie, a następnie przy pomocy funkcji *raise* wysyła sygnał do samego siebie oraz wykonuje odpowiednie dla danej opcji działanie, po czym tworzy potomka funkcją *fork* i ponownie przy pomocy funkcji *raise* potomek wysyła sygnał do samego siebie (z wyjątkiem opcji *pending*, gdzie testowane jest sprawdzenie, czy sygnał czekający w przodku jest widoczny w potomku).
- *Dziedziczenie ustawień sygnałów po wykonaniu funkcji exec.* W podobny sposób sprawdź jaki wpływ na ustawienia sygnałów ma wywołanie funkcji *exec*. Rozpatrz opcje: *ignore*, *mask* i *pending*.
- Przygotuj plik raport2.txt w którym nastąpi podsumowanie z wnioskami z wykonanych powyższych eksperymentów

Zadanie 2 (20%)

Przetestuj działanie trzech wybranych flag w funkcji *sigaction*. Jedną z nich powinna być flaga *SA_SIGINFO*. Dla tej flagi zainstaluj procedurę obsługi sygnału (handler) dla odpowiednio dobranych sygnałów stosując składnię procedury handlera z trzema argumentami. Wypisz i skomentuj (przygotowując odpowiednie scenariusze) trzy różne informacje, a dodatkowo także numer sygnału oraz identyfikator PID procesu wysyłającego dostarczane w strukturze *siginfo_t* przekazywanej jako drugi argument funkcji handlera.

Zadanie 3 (50%)

Napisz dwa programy: sender program wysyłający sygnały SIGUSR1 i catcher - program odbierający te sygnały. Program catcher jest uruchamiany jako pierwszy, wypisuje swój numer PID i czeka na sygnały SIGUSR1. Po każdorazowym odebraniu sygnału SIGUSR1 przez program catcher powinno nastąpić potwierdzenie odbioru tego sygnału. W tym celu, catcher wysyła do sendera sygnał SIGUSR1 informujący o odbiorze sygnału. Sender powinien wysłać kolejny sygnał dopiero po uzyskaniu tego potwierdzenia. Czekanie na takie potwierdzenie może odbywać się wywołując funkcję *sigsuspend*. Wraz z każdym sygnałem przesłanym przez sender powinien zostać przesłany tryb pracy programu catcher za pomocą funkcji *sigqueue*. Możliwe tryby pracy:

- 1- wypisanie na standardowym wyjściu liczb od 1 do 100
- 2- wypisanie na standardowym wyjściu aktualnego czasu
- 3 - wypisanie na standardowym wyjściu liczby otrzymanych żądań zmiany trybu pracy od początku działania programu
- 4 - wypisywanie na standardowym wyjściu aktualnego czasu co jedną sekundę aż do momentu otrzymania innego trybu pracy
- 5 – zakończenie działania programu catcher.

PID sendera catcher pobiera ze struktury *siginfo_t* po przechwyceniu od niego sygnału. Program sender jako pierwszy parametr przyjmuje PID procesu catcher. Następne parametry określają tryby pracy procesu catcher - w jednym wywołaniu może być przekazany jeden taki tryb lub więcej. Przy większej liczbie wywołanych trybów, powinny być one przekazane w kolejności do realizacji przez catcher. Program catcher działa aż do momentu otrzymania trybu 5 (zakończenie działania programu catcher). Program sender działa do momentu otrzymania potwierdzenia otrzymania przez catcher wszystkich przesłanych trybów, po czym kończy pracę. Program sender można wywołać wielokrotnie aż do zakończenia pracy przez catcher.

UWAGA! W żaden sposób nie opóźniamy wysyłania sygnałów, wszelkie "gubienie" sygnałów jest zjawiskiem naturalnym.