

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 działania, 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 składnie 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 zliczający ilość odebranych sygnałów. Ilość sygnałów *SIGUSR1* wysyłanych przez pierwszy program powinna być określana w parametrze wywołania tego programu. Program *catcher* jest uruchamiany najpierw, wypisuje swój numer PID i czeka na sygnały *SIGUSR1* i *SIGUSR2*. Wszystkie pozostałe sygnały są blokowane. Przyjmijmy, że czekanie na sygnały w poszczególnych procesach (w zależności od zadań) odbywa się wywołując funkcję *sigsuspend*. Program *sender* przyjmuje trzy parametry: PID procesu *catcher*, ilość sygnałów do wysłania i tryb wysłania sygnałów.

Po transmisji wszystkich sygnałów *SIGUSR1* *sender* powinien wysłać sygnał *SIGUSR2*, po otrzymaniu którego *catcher* wysyła do *sendera* tyle sygnałów *SIGUSR1*, ile sam ich otrzymał a „transmisję” kończy wysłaniem sygnału *SIGUSR2*, wypisaniem liczby odebranych sygnałów i zakończeniem działania. PID *sendera* *catcher* pobiera ze struktury *siginfo_t* po przechwyceniu od niego sygnału. Program *sender* po otrzymaniu sygnału *SIGUSR2* wyświetla komunikat o ilości otrzymanych sygnałów *SIGUSR1* oraz o tym, ile powinien ich otrzymać i kończy działanie.

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

a) Wysyłanie sygnałów w obu programach należy wykonać w następujących trybach: (30%)

- KILL - za pomocą funkcji *kill*
- SIGQUEUE - za pomocą funkcji *sigqueue* - wraz z przesłanym sygnałem *catcher* wysyła numer kolejnego odsyłanego sygnału, dzięki czemu *sender* wie, ile dokładnie *catcher* odebrał, a tym samym wysłał do niego sygnałów. Wypisz tę dodatkową informację w *senderze*.
- SIGRT - zastępując *SIGUSR1* i *SIGUSR2* dwoma dowolnymi sygnałami czasu rzeczywistego wysyłanymi za pomocą *kill*. Jaka liczba sygnałów będzie teraz odebrana?

b) Zmodyfikuj powyższe programy, dodając potwierdzenie odbioru sygnału po każdorazowym ich odebraniu przez program *catcher*. W tym celu, *catcher* wysyła do *sendera* sygnał *SIGUSR1* informujący o odbiorze sygnału. *Sender* powinien wysłać kolejny sygnał dopiero po uzyskaniu tego potwierdzenia. Zapewnij rozwiązanie, w którym ilość sygnałów odebranych jest zgodna z ilością sygnałów wysyłanych, i w którym nie dochodzi do zakleszczenia. (20%)