NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
ISLAMABAD

OPERATING SYSTEMS LAB SPRING 2022

Lab Manual 09
Signals Set

1 WHAT IS SIGNALS SET?

A signal set is a collection of one or more signals just like we define a set in mathematics.

1. Need a data type to represent multiple signals.
2. The sigset_t data type is used to represent a signal set.

```
sigset_t myset
```

2 FUNCTIONS TO MANIPULATE SIGNAL SETS

Following are the functions to manipulate signal sets:

```
Sigemptyset(sigset_t *set)
```

- Empty a signal set
- e.g. sigemptyset(&mysignals);

```
Sigfillset(sigset_t *set)
```

- Initialize a signal set with all signals
- e.g. sigfillset(&mysignals);

```
Sigaddset(sigset_t *set, int signo)
```

- Add sig.no into the specified set
- e.g.sigaddset(&mysignals,SIGUSR1)

```
Sigdelset(sigset_t *set, int signo)
```

- Delete signo from the specified set
- e.g. sigdelset(&mysignals,SIGUSR1)

```
Sigismember(sigset_t *set, int signo)
```

- Returns True if sig.no is the member of the set specified in sigset_t.
- e.g. sigismember(&mysignals,SIGINT)

3 EXAMPLE: MANIPULATING SIGNAL SETS

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <errno.h>

int main(){
        sigset_t sigset;
        // fill sigset with all available OS signals
        sigfillset(&sigset);
        /* Checking for membership*/
        if (sigismember(&sigset, SIGINT))
                printf(" SIGINT");
        if (sigismember(&sigset, SIGQUIT))
                printf(" SIGQUIT");
        if (sigismember(&sigset, SIGUSR1))
                printf(" SIGUSR1");
        if (sigismember(&sigset, SIGALRM))
                printf(" SIGALRM\n");

        /* Deleting SIGUSR1 from sigset */
        sigdelset(&sigset,SIGUSR1);
        if (sigismember(&sigset, SIGUSR1))
                printf("SIGUSR1\n");
        else
                printf("SIGUSR1 is Now not member of sigset \n");

              /* Empty the sigset */
        sigemptyset(&sigset);
        if (sigismember(&sigset, SIGALRM))
                printf(" SIGALRM");
        else
                printf("Sigset is empty\n");

        return 0;
}
```

4 SIGPROCMASK FUNCTION

Each process has a signal mask that defines the set of signals currently blocked from delivery to that process.
A process can:

1. Examine its signal mask
2. Change its signal mask
3. Perform both operations in one step by calling the following function:

```
int sigprocmask(int how, const sigset_t *set, sigset_t *oset);
```

FUNCTION PARAMETERS

1. **const sigset_t *set**
   New signal mask of sigset_t type
2. **sigset_t *oset**
   Old signal mask of sigset_t type
3. **int how**
   (a) **SIG_BLOCK** The set of blocked signals is the union of the current set and the set argument.
   (b) **SIG_UNBLOCK** The signals in set are removed from the current set of blocked signals. It is legal to attempt to unblock a signal which is not blocked.
   (c) **SIG_SETMASK** The set of blocked signals is set to the argument set.

5 EXAMPLE: SIGPROCMASK FUNCTION

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include<unistd.h>
#include<sys/wait.h>
int main()
{
        // new_mask set to store different signals
         sigset_t new_mask;
         // Initilize and Empty new_mask set
         sigemptyset(&new_mask);
        // Adding SIGQUIT to set named as new_mask
         sigaddset(&new_mask,SIGQUIT);
        // Adding SIGTSTP to set named as new_mask
        sigaddset(&new_mask,SIGTSTP);
        // Adding SIGINT to set named as new_mask
        sigaddset(&new_mask,SIGINT);
```

```
        /*
                * Block All the signal present in new_mask set.
        */
        sigprocmask(SIG_BLOCK,&new_mask,NULL);
         int i=0;
        for(i;; i++){
                printf("\n %d \n",i);
                        printf("My PID is %d\n",getpid());
                        printf("you can't close me using conventional
                                                                commands\n");
                        sleep(2);
          }
        return 0;
}
```

6 SIGPENDING FUNCTION

```
int sigpending(sigset_t *set);
```

The sigpending function returns the set of signals that are blocked from delivery and currently pending for the calling process. The set of signals is returned through the set argument.

EXAMPLE

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
void sig_quit(int signo){
        printf("caught SIGQUIT\n");
        signal(SIGQUIT, SIG_DFL); //Reseting the signal handler for SIGQUIT
}
void sig_term(int signo){
        printf("caught SIGINT\n");
        signal(SIGINT, SIG_DFL); //Reseting the signal handler for SIGINT
}
int main(void){
        sigset_t newmask, oldmask, pendmask;
        signal(SIGQUIT, sig_quit);
        signal(SIGINT, sig_term);

        //Block SIGQUIT and save current signal mask.
        sigemptyset(&newmask);
        sigaddset(&newmask, SIGQUIT);
        sigaddset(&newmask, SIGINT);

        // apply "newmask set" as mask for process and
        store old mask to "oldmask set"
        sigprocmask(SIG_BLOCK, &newmask, &oldmask);
        printf("Sleep for 10 sec\n");
```

```
        sleep(10);

        // SIGQUIT && SIGINT here will remain pending
        sigpending(&pendmask);
        if (sigismember(&pendmask, SIGQUIT) && sigismember(&pendmask, SIGINT))
                printf("\nSIGQUIT && SIGINT pending\n");
        // Restore signal mask which unblocks SIGQUIT && SIGINT.
        //and This will unmask the process and deliver all pending sig
        sigprocmask(SIG_SETMASK, &oldmask, NULL) ;
        printf("SIGQUIT && SIGINT unblocked\n");

        while(1){}
        exit(0);
        /* SIGQUIT here will terminate with core file */
}
```

7 SIGSUSPEND FUNCTION

Suspend the process untill any signal is received except sigmask.

```
int sigsuspend(const sigset_t *sigmask);
```

1. The signal mask of the process is set to the value pointed to by sigmask.
2. Then the process is suspended until a signal is caught or until a signal occurs that terminates the process.
3. If a signal is caught and if the signal handler returns, then sigsuspend returns, and the signal mask of the process is set to its value before the call to sigsuspend.

EXAMPLE

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

void sig_int(int signo){
        printf("\nin sig_int:\n ");
}

void sig_suspend(int signo){
        printf("\nin sig handler with %d number:\n ",signo);
}

int main(){
        sigset_t newmask, oldmask, waitmask;
        printf("program start: with pid %d \n",getpid());
        signal(SIGINT, sig_int);
        signal(SIGALRM, sig_suspend);
```

```
        signal(SIGUSR1, sig_suspend);
        sigemptyset(&waitmask);
        sigaddset(&waitmask, SIGUSR1);
        sigaddset(&waitmask, SIGALRM);
        sigemptyset(&newmask);
        sigaddset(&newmask, SIGINT);
        /*
                * Block SIGINT and save current signal mask.
        */

        sigprocmask(SIG_BLOCK, &newmask, &oldmask);

        /*
        * Critical region of code.
        */

        printf("in critical region: \n");

        /*
                * It will Pause untill received any signal
                except SIGUSR1 & SIGALRM.
                * newmask set will be suspended for that period of time
        */

        sigsuspend(&waitmask);

         // Now onwards old mask "newmask" is applicable
        printf("after return from sigsuspend: \n");
        sleep(15);
        /*
                        * unblocking newmask set.
        */
        sigprocmask(SIG_UNBLOCK, &newmask, NULL);

        /*
        * And continue processing ...
        */
        printf("program exit: \n");
        exit(0);
}
```