Linux Signal Handling Interview Questions And Answers Guide.

Global Guideline.

https://globalguideline.com/

Linux Signal Handling Job Interview Preparation Guide.

Question # 1

Which signal is generated when we press control-C?
a) SIGINT
b) SIGTERM
c) SIGKILL
d) SIGSEGV

Answer:-

a) SIGINT

Read More Answers.

Question # 2

If a signal is received by a process, when will it be processed?
a) It is processed immediately
b) It is processed when process is switching to kernel mode
c) It is processsed in the next timeslice given to the process

Answer:-

b) It is processed when process is switching to kernel mode

Read More Answers.

Question # 3

Which signal is generated when we press ctrl-Z?
a) SIGKILL
b) SIGSTOP
c) SIGABRT
d) SIGINT

Answer:-

d) SIGINT

Read More Answers.

Question # 4

Which signal is sent when the Child process terminates?
a) SIGINIT
b) SIGKILL
c) SIGSTOP
d) SIGCHLD

Answer:-

b) SIGKILL

Read More Answers.

Question # 5

Which of the following signal cannot be handled or ignored?
a) SIGINT
b) SIGCHLD
c) SIGKILL
d) SIGALRM

Answer:-

c) SIGKILL

Read More Answers.

Question # 6

What happnes as the signal SIGINT hits the current process in the program?

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

void response (int);
void response (int sig_no)
{
   printf("Linuxn");
}
int main()
{
   struct sigaction act;
   act.sa_handler = response;
   act.sa_flags = 0;
   sigemptyset(&act.sa_mask);
   sigaction(SIGINT,&act,0);
   while(1){
      printf("googlen");
      sleep(1);
   }
   return 0;
}
```

a) the process terminates
b) the string "Linux" prints
c) the string "Linux" prints and then process terminates
d) none of the mentioned

Answer:-

b) the string "Linux" prints
Output:

```
[root@localhost sigaction]# gcc -o san san.c
[root@localhost sigaction]# ./san
google
google
google
^CLinux
google
google
^CLinux
google
^Z
[7]+ Stopped ./san
[root@localhost google]#
```

Read More Answers.

Question # 7

This program will print:

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

void response (int);
void response (int sig_no)
{
   printf("%s is workingn",sys_siglist[sig_no]);
}
int main()
{
   alarm(5);
   sleep(50);
   printf("googlen");
   signal(SIGALRM,response);
   return 0;
}
```

a) "google"
b) "Alarm clock"
c) nothing
d) none of the mentioned

Answer:-

b) "Alarm clock"
Explanation:After 5 seconds of the execution of this program, the signal SIGALRM hits the process and handler executes.
Output:

```
[root@localhost google]# gcc -o san san.c
[root@localhost google]# ./san
Alarm clock
[root@localhost google]#
```

Read More Answers.

Question # 8

What is the output of this program?

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

void response (int);
void response (int sig_no)
{
   printf("%sn",sys_siglist[sig_no]);
   printf("This is singal handlern");
}
int main()
{
   pid_t child;
   int status;
   child = fork();
   switch (child){
      case -1 :
         perror("fork");
         exit (1);
      case 0 :
         kill(getppid(),SIGKILL);
         printf("I am an orphan process because my parent has been killed by men");
         printf("Handler failedn");
         break;
      default :
         signal(SIGKILL,response);
         wait(&status);
         printf("The parent process is still aliven");
         break;
   }
   return 0;
}
```

a) the child process kills the parent process
b) the parent process kills the child process
c) handler function executes as the signal arrives to the parent process
d) none of the mentioned

Answer:-

a) the child process kills the parent process
Explanation:
The SIGKILL signal can not be handled by singal handler function.
Output:

```
[root@localhost google]# gcc -o san san.c
[root@localhost google]# ./san
Killed
[root@localhost google]# I am an orphan process because my parent has been killed by me
Handler failed
[root@localhost google]#
```

Read More Answers.

Question # 9

Which one of the following is not true about this program?

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

void response (int);
void response (int signo)
{
   printf("%sn",sys_siglist[signo]);
   signal(SIGSEGV,SIG_IGN);
}
int main()
{
   signal (SIGSEGV,response);
   char *str;
   *str = 10;
   return 0;
}
```

a) kernel sends SIGSEGV signal to a process as segmentation fault occurs
b) in this process signal handler will execute only one time of recieving the signal SIGSEGV
c) both (a) and (b)
d) none of the mentioned

Answer:-

d) none of the mentioned
Explanation:
In this process the segmentation fault occurs because the memory is not allocated to the pointer *str.
Output:

```
[root@localhost google]# gcc -o san san.c
```

```
[root@localhost google]# ./san
Segmentation fault
Segmentation fault (core dumped)
[root@localhost google]#
```

Read More Answers.

Question # 10

What is the output of this program?

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

void response (int);
void response (int sig_no)
{
   printf("%sn",sys_siglist[sig_no]);
}
int main()
{
   pid_t child;
   int status;
   child = fork();
   switch(child){
      case -1:
         perror("fork");
      case 0:
         break;
      default :
         signal(SIGCHLD,response);
         wait(&status);
         break;
   }
}
```

a) this program will print nothing
b) this program will print "Child Exited"
c) segmentation fault
d) none of the mentioned

Answer:-

b) this program will print "Child Exited"
Explanation:
The child process sends SIGCHILD signal to its parent as it terminates.
Output:

```
[root@localhost google]# gcc -o san san.c
[root@localhost google]# ./san
Child exited
[root@localhost google]#
```

Read More Answers.

Question # 11

In this program

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

int main()
{
   pid_t child;
   child=fork();
   switch(child){
      case -1 :
         perror("fork");
         exit(1);
      case 0 :
         while(1){
            printf("Child Processn");
            sleep(1);
         }
         break;
      default :
         sleep(5);
         kill(child,SIGINT);
         printf("The child process has been killed by the parent processn");
         break;
   }
   return 0;
}
```

a) the child process kills the parent process
b) the parent process kills the child process
c) both the processes are killed by each other
d) none of the mentioned

Answer:-

b) the parent process kills the child process
Explanation:
The parnet process kills the child by sending a signal.
Output:

```
[root@localhost google]# gcc -o san san.c
[root@localhost google]# ./san
Child Process
Child Process
Child Process
Child Process
Child Process
The child process has been killed by the parent process
[root@localhost google]#
```

Read More Answers.

Question # 12

What will print as the SIGINT signal hits the running process of this program?

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

void response (int);
void response (int sig_no)
{
   printf("%s",sys_siglist[sig_no]);
}
int main()
{
   signal(SIGINT,response);
   while(1){
      printf("googlen");
      sleep(1);
   }
   return 0;
}
```

a) Interrupt
b) Stop
c) Terminate
d) none of the mentioned

Answer:-

a) Interrupt
Explanation:
The messages associated with signals can be access by the function sys_siglist().
Output:

```
[root@localhost google]# gcc -o san san.c
[root@localhost google]# ./san
google
google
google
^CInterruptgoogle
google
^CInterruptgoogle
google
^CInterruptgoogle
google
google
^Z
[4]+ Stopped ./san
[root@localhost google]#
```

Read More Answers.

Question # 13

What happens as the SIGINT signal hits the running process of this program?

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

int main()
{
   pid_t child;
   signal(SIGINT,SIG_IGN);
   child=fork();
   switch(child){
      case -1:
         perror("fork");
         exit(1);
      case 0:
         while(1){
```

```
            printf("Child Processn");
            sleep(1);
          }
          break;
        default :
          while(1){
            printf("Parent Processn");
            pause();
          }
          break;
    }
    return 0;
  }
```

a) child process terminates
b) parent process terminates
c) both child and parent process ignores the signal
d) none of the mentioned

Answer:-

c) both child and parent process ignores the signal
Explanation:
If a process ignores a signal then by default its child also ignores that signal.
Output:

```
[root@localhost google]# gcc -o san san.c
[root@localhost google]# ./san
Parent Process
Child Process
Child Process
^CChild Process
^CChild Process
^CChild Process
^Z
[3]+ Stopped ./san
[root@localhost signal]#
```

Read More Answers.

Question # 14

What will happen if we press "Ctrl+c" key two times after running this program?

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

  void response(int);
  void response(int sig_no)
  {
    printf("Linuxn");
    signal(SIGINT,SIG_DFL);
  }
  int main()
  {
    signal(SIGINT,response);
    while(1){
      printf("googlen");
      sleep(1);
    }
    return 0;
  }
```

a) process will terminate in the first time
b) process will terminate in the second time
c) process will never terminate
d) none of the mentioned

Answer:-

c) process will never terminate
Explanation:
According to the signal handler function of this program as the SIGINT signal arrives second time, the signal performs its default operation i.e. termination of the process.
Output:

```
[root@localhost google]# gcc -o san san.c
[root@localhost google]# ./san
google
google
^CLinux
google
^C
[root@localhost google]#
```

Read More Answers.

Question # 15

What will happen as we press the "Ctrl+c" key after running this program?

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

```
#include<signal.h>

void response (int);
void response (int sig_no)
{
   printf("Linuxn");
}
int main()
{
   signal(SIGINT,response);
   while(1){
      printf("googlen");
      sleep(1);
   }
   return 0;
}
```

a) the string "Linux" will print
b) the process will be terminated after printing the string "Linux"
c) the process will terminate
d) none of the mentioned

Answer:-

a) the string "Linux" will print
Explanation:
The signal handler function "response" executes after recieving the signal SIGINT.
Output:

```
[root@localhost google]# gcc -o san san.c
[root@localhost google]# ./san
google
google
google
^CLinux
google
google
^CLinux
google
google
^CLinux
google
^Z
[2]+ Stopped ./san
[root@localhost google]#
```

Read More Answers.

Question # 16

Another signal that cannot be caught is:
a) SIGPIPE
b) SIGHUP
c) SIGSTOP
d) SIGUSR1

Answer:-

c) SIGSTOP

Read More Answers.

Question # 17

When real interval timer expires which signal is generated?
a) SIGINT
b) SIGCHLD
c) SIGKILL
d) SIGALRM

Answer:-

d) SIGALRM

Read More Answers.

Question # 18

Signals are handled using which system call?
a) kill
b) signal
c) both
d) none

Answer:-

b) signal

Read More Answers.

Question # 19

Default action of SIGSEGV is:
a) Terminate
b) Core dump
c) Stop
d) Cont

Answer:-

b) Core dump

Read More Answers.

Question # 20

The kill system call is used to:
a) Send shutdown messages to all by superuser
b) Send a signal to a process
c) Kill processes
d) Stop the processes

Answer:-

b) Send a signal to a process

Read More Answers.

Question # 21

What is the output of the below code?

```
void sig_handler ( int signum) {
   printf("Handled the signaln");
}
int main() {
   int pid;
   signal (SIGKILL, sig_handler);
   pid = fork();
   if (pid==0) {
      kill(getppid(), SIGKILL);
      exit(0);
   } else {
      sleep(20);
   }
   return 0;
}
```

a) Error child cannot send a SIGKILL signal to parent.
b) Parent goes to the signal handler, prints handled the signal and goes back to sleep
c) Parent goes to the signal handler, prints handled the signal and exits
d) Parent exits without going to the signal handler

Answer:-

d) Parent exits without going to the signal handler

Read More Answers.

Operating System Linux Most Popular Interview Topics.

1 : Linux OS Frequently Asked Interview Questions and Answers Guide.

2 : Linux Commands Frequently Asked Interview Questions and Answers Guide.

3 : Linux IPC Frequently Asked Interview Questions and Answers Guide.

4 : Linux General Frequently Asked Interview Questions and Answers Guide.

5 : Linux System Calls Frequently Asked Interview Questions and Answers Guide.

6 : Linux Device Drivers Frequently Asked Interview Questions and Answers Guide.

7 : Linux Socket Programming Frequently Asked Interview Questions and Answers Guide.

8 : Linux Threads Frequently Asked Interview Questions and Answers Guide.

9 : Makefile Frequently Asked Interview Questions and Answers Guide.

10 : Awk Programming Frequently Asked Interview Questions and Answers Guide.

About Global Guideline.

Global Guideline is a platform to develop your own skills with thousands of job interview questions and web tutorials for fresher's and experienced candidates. These interview questions and web tutorials will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts. Global Guideline invite you to unlock your potentials with thousands of **Interview Questions with Answers** and much more. Learn the most common technologies at Global Guideline. We will help you to explore the resources of the World Wide Web and develop your own skills from the basics to the advanced. Here you will learn anything quite easily and you will really enjoy while learning. Global Guideline will help you to become a professional and Expert, well prepared for the future.

* This PDF was generated from https://GlobalGuideline.com at **November 29th, 2023**

* If any answer or question is incorrect or inappropriate or you have correct answer or you found any problem in this document then don't hesitate feel free and e-mail us we will fix it.

You can follow us on FaceBook for latest Jobs, Updates and other interviews material.
www.facebook.com/InterviewQuestionsAnswers

Follow us on Twitter for latest Jobs and interview preparation guides
https://twitter.com/InterviewGuide

Best Of Luck.

Global Guideline Team
https://GlobalGuideline.com
Info@globalguideline.com