# Important marks:

*(incomplete) -*  the answers marked as correct are NOT ALL correct answers.

answer - correct answer

answer - wrong answer

# answer - you *think* it’s a correct answer

CAPS or callout - an important note about the task.

YOU ARE ALLOWED TO ADD/UPDATE/CORRECT ANSWERS AND QUESTIONS IF YOU ARE SURE. IF YOU ARE NOT - USE YELLOW.

# 

# 

# Which functions are used to communicate with named pipe (entire lifetime of the pipe)?

TEMP\_FAILURE\_RETRY

close

read

open

write

Pipe

# Which functions are used to communicate with named pipe (entire lifetime of a pipe)?

Mkfifo

Read

Isalpha

Write

Fifo

# Which functions are used to communicate with unnamed pipe that exists ?

Read

Open

Isalpha

Write

Fifo

Mkpipe

# Which functions are used to communicate with unnamed pipe (entire lifetime of the pipe)?

TEMP\_FAILURE\_RETRY

close

read

open

write

Pipe

# Which of these describe unnamed pipe?

is created by mkpipe function

requires only one unique file descriptor within the filesystem

may be used only by related processes

provides half duplex data flow

**Which of these describe unnamed pipe?**

is created by pipe function

can be referred to by name

is a file

Enforces to create file descriptors for both reading and writing

**Which of these describe unnamed pipe? (incomplete)**

Exists in filesystem

Allows to define own file descriptors when creating

is created by mkfifo function

Enables inter-process communication

# Main process calls the following functions in sequence: fork, pipe, fork, fork, fork. How many processes would have access to the pipe created by main process?

3

4

7

8

16

32

-> Version “fork, pipe, fork, fork”: 4

-> Version “fork, fork, pipe, fork”: 2

# Write function:

-in nonblocking mode, if there is no space in pipe to write any byte, fails with errno set to EAGAIN

-in blocking mode, blocks the calling thread until it writes all the data or until the pipe is full

-in blocking mode, if size of the message is smaller than PIPE\_BUF but there is no enough space in the pipe to wirte the whole message, the function writes part of the message until the pipe is full and waits to write the rest

-by default runs in nonblocking mode (I think)

-enables atomic write to a pipe if size of the message is less or equal PIPE\_BUF

# Write function:

* in blocking mode, if size of the message is smaller than PIPE\_BUF but there is no enough space in the pipe to write the whole message, THE FUNCTION WAITS
* reading from pipe can be run in blocking mode by opening the pipe with an appropriate flag
* enables atomic write to a pipe if size of the message is greater than PIPE\_BUF
* in nonblocking mode, if there is space to write only part of the message to the pipe, it will always write it and return number of written bytes

# Write function:

in nonblocking mode, if all reading processes are already disconnected, fails with ENXIO error

always enables atomic write to a pipe

in blocking mode, blocks the calling thread until it writes all the data, regardless of its size, to the pipe

by default runs in blocking mode

# Which of these describe unnamed pipe?

Provides half-duplex data flow

May be used only by related processes

Requires only one unique file desciptor within the filesystem

Is created by mkpipe function

Can be referred to by file descriptor

# Which of these describe unnamed pipe?

* in order to use a new one, a programmer must remember about creating and then opening it
* may be created by open function
* allows to create only one file descriptor only for reading
* Can be referred to by file descriptor

**Which of these describe unnamed pipe?**

* is created by fifo function
* provides simplex data flow
* has a pair of unique file descriptors within the process
* allows to create only one file descriptor only for writing

# Main process creates an unnamed pipe and the calls fork() 2 times. How many file descriptors in total should be closed?

3

4

8

16

5

9

# Main process creates an unnamed pipe and the calls fork() 3 times. How many file descriptors in total should be closed?

* 4
* 3
* 7
* 8
* 16
* 32,ii,ii

# Which functions may be used to create file descriptor to the pipe (named or unnamed)?

* Open
* Mkfifo
* Fifo
* Read
* Write
* Pipe

# 

# A process has established bi-directional communication with its child process using pipes. How many times does it have to use close() to release all link-related resources (descriptors)?

0

1

2

3

4

# 

# What is true about mkfifo():

* It creates a link opened for both reading and writing
* Returns 2 descriptors in an array provided as an argument
* Creates a link that is yet to be opened
* It sends an array of data through the link

# Unnamed links allow for communication:

* Between parent and child processes
* Between 2 unrelated processes
* Between 2 processes running on 2 different machines (computers)
* Within single process (the same process reads and writes)

# What are the differences between named and unnamed links (pipes and fifos)

* Fifo can have both ends opened by the same process while pipe can’t
* Fifo is visible as a special file in the filesystem
* Fifo can be used in nonblocking mode while pipe can’t

# How is process informed about broken link (disconnection of the other end)

* Process is not informed
* Process receives signal SIGPIPE when the other process closes its end
* read() returns 0
* read() returns -1 setting error to EPIPE
* Process receives signal SIGPIPE when trying to use read()

# What’s true about pipe()?

* Creates unnamed link
* Creates named link
* Returns file descriptor used for reading
* Returns file descriptor used for writing

# Process can use unnamed link to communicate with:

* Any process on the same device
* Its children
* Children of its children
* Any process in the system

# How many links are needed for two way communication between parent process and its child?

* 2
* 1
* 4
* It’s not possible

**Write function:**

* In nonblocking mode, if there is space to write only part of the message, depending on the message length, it may not write a byte and return -1
* Enables atomic write to a pipe if size of the message equals 5 kB
* In blocking mode, if size of the message is smaller than PIPE\_BUF but there is not enough space in the pipe to write the whole message, the function writes part of the message until the pipe is full and waits to write the rest
* Running in nonblocking mode is implemented by creating a thread in the background

# 

**Which of these describe unnamed pipe?**

* allows to define own file descriptors when creating
* enables inter-process communication
* is created by mkfifo function
* exists in the filesystem