Problem and answer

|  |
| --- |
| Problem: Client can receive mjpeg streaming no problem, however, when client disconnected, my program can produce error and crash |
| Answer:  It seems the problem might occurs inside the write\_multipart\_header(), inside write\_to\_client(). It seems error inside linux function write(..)  What causes "[Errno 32] Broken pipe" in Python?  "Broken pipe" is essentially an IOError error (short for input/output error), which happened at the Linux system level. It usually occurs when reading and writing files, or in other words, doing file input/output or network input/output (via sockets).  The corresponding Linux system error is EPIPE, excerpted from GNU libc error codes:   |  | | --- | | *Macro:*int***EPIPE***  *“Broken pipe.” There is no process reading from the other end of a pipe. Every library function that returns this error code also generates a SIGPIPE signal; this signal terminates the program if not handled or blocked. Thus, your program will never actually see EPIPE unless it has handled or blocked SIGPIPE.* |   we know that [Errno 32] Broken pipe is caused by the system sending SIGPIPE signal, which is an inter-process communication mechanism of Linux.  For example, SIGINT is another signal used internally by Linux system. In Linux, Ctrl+C will send a SIGINT signal to end the process, or we can use the kill command to achieve the same effect.  Python does not ignore SIGPIPE by default. Instead, it translates the signal into an exception and raises IOError: [Errno 32] Broken pipe every time it receives a SIGPIPE.  [Errno 32] Broken pipe when pipe outputs in Linux terminal  If you encounter [Errno 32] Broken pipe when trying to pipe output of a Python script to another program such as the below example, read on.   |  | | --- | | python <filename>.py | head |   This pipeline syntax will create a process that sends data upstream, and a process that reads data downstream. When the downstream does not need to read upstream data, it will send a SIGPIPE signal to the upstream process.  When downstream no longer needs to read upstream data? For example, the head command in the example only needs to read enough lines to tell the upstream that I no longer need to read it, and it will send the SIGPIPE signal to the upstream process.  When the upstream process is a Python program, an error such as **IOError: [Errno 32] Broken pipe** will occur.  **Avoid [Errno 32] Broken pipe by ignoring SIGPIPE**  If you don't care too much about properly catching SIGPIPE and just need to get things running quickly, add the code snippet below to the top of your Python program.   |  | | --- | | from signal import signal, SIGPIPE, SIG\_DFL  #Ignore SIG\_PIPE and don't throw exceptions on it... (http://docs.python.org/library/signal.html)  signal(SIGPIPE,SIG\_DFL) |   What the code does is redirecting SIGPIPE signals to the default SIG\_DFL, which the system usually ignore.  But beware, the Python manual on signal library warn against this type of handling SIGPIPE   |  | | --- | | *Do not set*[***SIGPIPE***](https://docs.python.org/3/library/signal.html#signal.SIGPIPE)*’s disposition to*[***SIG\_DFL***](https://docs.python.org/3/library/signal.html#signal.SIG_DFL)*in order to avoid****[BrokenPipeError](https://docs.python.org/3/library/exceptions.html" \l "BrokenPipeError)****. Doing that would cause your program to exit unexpectedly also whenever any socket connection is interrupted while your program is still writing to it.* |   **Properly catch IOError to avoid [Errno 32] Broken pipe**  Since [Errno 32] Broken pipe is actually a IOError, you can place a try/catch block to catch it like the code snippet below :   |  | | --- | | import sys, errno  try:  ### IO operation ###  except IOError as e:  if e.errno == errno.EPIPE:  ### Handle the error ### |   **Possible solution for [Errno 32] Broken pipe in multi-process program.**  In programs that use worker processes to speed up processing and make use of multi-core CPUs, you can try reducing the number of the worker processes to see whether the error disappear or not.  A large number of worker processes may conflict with each other when they try to take control of system resources or the permission to write into disk.  Code          Normal one should be like the following    In python file I add try catch and get following     |  | | --- | | close failed in file object destructor:  sys.excepthook is missing  lost sys.stderr |   Why after cleanup, still try to send multiheader. Maybe need to look into cleanup\_client    The following is how the original streameye should do after cleanup\_client  At very first client browser connect, it will directly connect with two port (so client number is now 2). After a while, read\_request cannot read one of client, so program will cleanup\_client. so client number is now 1    The following is I close browser immediately I get connected, didn’t wait for first client being removed by program after timeout.    The following is I close browser after waiting for first client being removed by program after timeout.    On the other hand, my code seems not to auto timeout the read\_request    Actually, I found out that I forgot to add the following code inside function wait\_for\_client so that we can set timeout for socket. The following code does shows up in streameye.c   |  | | --- | | /\* set socket timeout \*/    struct timeval tv;  tv.tv\_sec = 1;  tv.tv\_usec = 0;  setsockopt(stream\_fd, SOL\_SOCKET, SO\_RCVTIMEO, (char \*) &tv, sizeof(struct timeval));  setsockopt(stream\_fd, SOL\_SOCKET, SO\_SNDTIMEO, (char \*) &tv, sizeof(struct timeval)); |   So after we add the above code, now my program will auto timeout the read\_request  I don’t know why sometime my code would work when client disconnected, but sometimes go into error.    I found that streameye.c will also suffer from broken pipe like me (after adding printf the error in streameye.c). But it seems it got error handle while I don’t. That is the problem I think      Finally, adding the signal part, problem fixed   |  | | --- | | #include <signal.h> //singal  /\*  #define INFO(fmt, ...) fprintf(stderr, "%s: INFO : " fmt "\n", str\_timestamp(), ##\_\_VA\_ARGS\_\_)  #define ERROR(fmt, ...) fprintf(stderr, "%s: ERROR: " fmt "\n", str\_timestamp(), ##\_\_VA\_ARGS\_\_)  #define ERRNO(msg) ERROR("%s: %s", msg, strerror(errno))  #define ERROR\_CLIENT(client, fmt, ...) ERROR("%s:%d: " fmt, client->addr, client->port, ##\_\_VA\_ARGS\_\_)  #define ERRNO\_CLIENT(client, msg) ERROR\_CLIENT(client, "%s: %s", msg, strerror(errno))  \*/  int main(int argc, char \*argv[]){  \*\*\*\*\*\*  /\* signals \*/  DEBUG("installing signal handlers");  struct sigaction act;  act.sa\_handler = bye\_handler;  act.sa\_flags = 0;  sigemptyset(&act.sa\_mask);  if (sigaction(SIGINT, &act, NULL) < 0) {  //ERRNO("sigaction() failed");  return -1;  }  if (sigaction(SIGTERM, &act, NULL) < 0) {  //ERRNO("sigaction() failed");  return -1;  }  if (signal(SIGPIPE, SIG\_IGN) == SIG\_ERR) {  //ERRNO("signal() failed");  return -1;  }  …….  }  void bye\_handler(int signal) {  if (!running) {  //INFO("interrupt already received, ignoring signal");  return;  }  //INFO("interrupt received, quitting");  running = 0;  }  char \*str\_timestamp() {  static \_\_thread char s[20];  time\_t t = time(NULL);  struct tm \*tmp = localtime(&t);  strftime(s, sizeof(s), "%Y-%m-%d %H:%M:%S", tmp);  return s;  } |     I later found that I don’t need to add so much code, I only need the following.   |  | | --- | | #include <signal.h> //singal  #include <errno.h>  if (signal(SIGPIPE, SIG\_IGN) == SIG\_ERR) {  //ERRNO("signal() failed");  return -1;  }  //above code means ignore SIGPIPE |   SIGPIPE is for situations like this:  Code:   |  | | --- | | $ grep "pattern" < reallyhugefile | head |   grep might print millions of lines, but head only reads 10 then quits. Once head closes the read-end and quits, grep gets SIGPIPE, which kills it, forcing it to quit early instead of processing the entire file uselessly.  If you don't want your program to be killed, handle or block SIGPIPE yourself. You will start getting write-errors with errno set to EPIPE instead.  seq | head -n 1  The command from above creates two processes, which are connected by a <man:pipe(2)>. seq writes its infinite sequence of numbers to STDOUT, while head reads the other end of the pipe as STDIN. It reads the first line and then exits. But what stops seq from running until the collapse of the universe?  The Linux kernel only allocates a finite sized buffer for that pipe. The size of that buffer changed over time from 4 KiB to 64 KiB to configurable, but still defaults to 1 MiB. See <man:pipe(7)> for more details about the getting the size.  After seq filled up that buffer its next call to <man:write(2)> will block until the reader has read some data and thus has freed some space in the buffer. But as soon as head terminated, there will never be any other reader who can do that. The Linux kernel thus sends SIGPIPE to seq to signal it, that no reader is left. The default action for that signal is terminate the process.  If the calling process is ignoring SIGPIPE, then <man:write(2)> fails with the error EPIPE. |
| Reference:  Broken pipe: <https://linuxpip.org/broken-pipe-python-error/>  close failed in file object destructor: <https://stackoverflow.com/questions/42722411/errors-at-python-program-exit-close-failed-in-file-object-destructor-sys-ex>  signal: <https://www.tutorialspoint.com/c_standard_library/c_function_signal.htm>  SIGPIPE and EPIPE : <https://www.unix.com/programming/171395-sigpipe-epipe.html>  SIGPIPE, EPIPE: <https://pmhahn.github.io/SIGPIPE/>  Why does SIGPIPE exist: <https://stackoverflow.com/questions/8369506/why-does-sigpipe-exist/9337925> |

|  |
| --- |
| Problem: Cannot find any user defined variable called errno |
| Answer:  1. The <errno.h> header file defines the integer variable errno, which is set by system calls and some library functions in the event of an error to indicate what went wrong.  2. The following is the list of output from errno -l  1 EPERM Operation not permitted  2 ENOENT No such file or directory  3 ESRCH No such process  4 EINTR Interrupted system call  5 EIO Input/output error  6 ENXIO No such device or address  7 E2BIG Argument list too long  8 ENOEXEC Exec format error  9 EBADF Bad file descriptor  10 ECHILD No child processes  11 EAGAIN Resource temporarily unavailable  11 EWOULDBLOCK Resource temporarily unavailable  12 ENOMEM Cannot allocate memory  13 EACCES Permission denied  14 EFAULT Bad address  15 ENOTBLK Block device required  16 EBUSY Device or resource busy  17 EEXIST File exists  18 EXDEV Invalid cross-device link  19 ENODEV No such device  20 ENOTDIR Not a directory  21 EISDIR Is a directory  22 EINVAL Invalid argument  23 ENFILE Too many open files in system  24 EMFILE Too many open files  25 ENOTTY Inappropriate ioctl for device  26 ETXTBSY Text file busy  27 EFBIG File too large  28 ENOSPC No space left on device  29 ESPIPE Illegal seek  30 EROFS Read-only file system  31 EMLINK Too many links  32 EPIPE Broken pipe  33 EDOM Numerical argument out of domain  34 ERANGE Numerical result out of range  35 EDEADLK Resource deadlock avoided  35 EDEADLOCK Resource deadlock avoided  36 ENAMETOOLONG File name too long  37 ENOLCK No locks available  38 ENOSYS Function not implemented  39 ENOTEMPTY Directory not empty  40 ELOOP Too many levels of symbolic links  42 ENOMSG No message of desired type  43 EIDRM Identifier removed  44 ECHRNG Channel number out of range  45 EL2NSYNC Level 2 not synchronized  46 EL3HLT Level 3 halted  47 EL3RST Level 3 reset  48 ELNRNG Link number out of range  49 EUNATCH Protocol driver not attached  50 ENOCSI No CSI structure available  51 EL2HLT Level 2 halted  52 EBADE Invalid exchange  53 EBADR Invalid request descriptor  54 EXFULL Exchange full  55 ENOANO No anode  56 EBADRQC Invalid request code  57 EBADSLT Invalid slot  59 EBFONT Bad font file format  60 ENOSTR Device not a stream  61 ENODATA No data available  62 ETIME Timer expired  63 ENOSR Out of streams resources  64 ENONET Machine is not on the network  65 ENOPKG Package not installed  66 EREMOTE Object is remote  67 ENOLINK Link has been severed  68 EADV Advertise error  69 ESRMNT Srmount error  70 ECOMM Communication error on send  71 EPROTO Protocol error  72 EMULTIHOP Multihop attempted  73 EDOTDOT RFS specific error  74 EBADMSG Bad message  75 EOVERFLOW Value too large for defined data type  76 ENOTUNIQ Name not unique on network  77 EBADFD File descriptor in bad state  78 EREMCHG Remote address changed  79 ELIBACC Can not access a needed shared library  80 ELIBBAD Accessing a corrupted shared library  81 ELIBSCN .lib section in a.out corrupted  82 ELIBMAX Attempting to link in too many shared libraries  83 ELIBEXEC Cannot exec a shared library directly  84 EILSEQ Invalid or incomplete multibyte or wide character  85 ERESTART Interrupted system call should be restarted  86 ESTRPIPE Streams pipe error  87 EUSERS Too many users  88 ENOTSOCK Socket operation on non-socket  89 EDESTADDRREQ Destination address required  90 EMSGSIZE Message too long  91 EPROTOTYPE Protocol wrong type for socket  92 ENOPROTOOPT Protocol not available  93 EPROTONOSUPPORT Protocol not supported  94 ESOCKTNOSUPPORT Socket type not supported  95 ENOTSUP Operation not supported  95 EOPNOTSUPP Operation not supported  96 EPFNOSUPPORT Protocol family not supported  97 EAFNOSUPPORT Address family not supported by protocol  98 EADDRINUSE Address already in use  99 EADDRNOTAVAIL Cannot assign requested address  100 ENETDOWN Network is down  101 ENETUNREACH Network is unreachable  102 ENETRESET Network dropped connection on reset  103 ECONNABORTED Software caused connection abort  104 ECONNRESET Connection reset by peer  105 ENOBUFS No buffer space available  106 EISCONN Transport endpoint is already connected  107 ENOTCONN Transport endpoint is not connected  108 ESHUTDOWN Cannot send after transport endpoint shutdown  109 ETOOMANYREFS Too many references: cannot splice  110 ETIMEDOUT Connection timed out  111 ECONNREFUSED Connection refused  112 EHOSTDOWN Host is down  113 EHOSTUNREACH No route to host  114 EALREADY Operation already in progress  115 EINPROGRESS Operation now in progress  116 ESTALE Stale file handle  117 EUCLEAN Structure needs cleaning  118 ENOTNAM Not a XENIX named type file  119 ENAVAIL No XENIX semaphores available  120 EISNAM Is a named type file  121 EREMOTEIO Remote I/O error  122 EDQUOT Disk quota exceeded  123 ENOMEDIUM No medium found  124 EMEDIUMTYPE Wrong medium type  125 ECANCELED Operation canceled  126 ENOKEY Required key not available  127 EKEYEXPIRED Key has expired  128 EKEYREVOKED Key has been revoked  129 EKEYREJECTED Key was rejected by service  130 EOWNERDEAD Owner died  131 ENOTRECOVERABLE State not recoverable  132 ERFKILL Operation not possible due to RF-kill  133 EHWPOISON Memory page has hardware error |
| Reference:  1. <https://stackoverflow.com/questions/503878/how-to-know-what-the-errno-means>  2. <https://man7.org/linux/man-pages/man3/errno.3.html> |

|  |
| --- |
| Problem: Error - undefined reference to 'pthread\_create' with C program in GCC Linux |
| Answer  1. Include Header file   |  | | --- | | #include <stdio.h>  #include <pthread.h> |   2. Compile command   |  | | --- | | gcc main.c -o main -lpthread | |
| Reference:  <https://www.includehelp.com/c-programming-questions/error-undefined-reference-to-pthread-create-in-linux.aspx> |

|  |
| --- |
| Problem:  1. My http server program stuck at accept function.  2. After accept client, reply with some jpeg data to client, my code stuck at pthread\_cond\_wait(…). It doesn’t go back to main thread to process the camera input data. |
| Answer:  *accept* is a blocking call unless you specify the socket to be nonblocking. You can achieve this with the following:   |  | | --- | | fcntl(sock\_desc, F\_SETFL, fcntl(sock\_desc, F\_GETFL, 0) | O\_NONBLOCK); |   You can do error checking with the return value from fcntl.  Actually I forgot to copy this part of code from streameye.c |
| Reference:  <https://stackoverflow.com/questions/30733924/server-program-gets-stuck-at-accept-function/30734811> |

|  |
| --- |
| Problem: Difference between pthread and fork on gnu/Linux |
| Answer:  In C there are some differences however:  fork()   * Purpose is to create a new process, which becomes the child process of the caller * Both processes will execute the next instruction following the fork() system call * Two identical copies of the computer's address space,code, and stack are created one for parent and child.   Thinking of the fork as it was a person; Forking causes a clone of your program (process), that is running the code it copied.  pthread\_create()   * Purpose is to create a new thread in the program which is given the same process of the caller * Threads within the same process can communicate using shared memory. (Be careful!) * The second thread will share data,open files, signal handlers and signal dispositions, current working directory, user and group ID's. The new thread will get its own stack, thread ID, and registers though.   Continuing the analogy; your program (process) grows a second arm when it creates a new thread, connected to the same brain. |
| Reference:  <https://stackoverflow.com/questions/5514464/difference-between-pthread-and-fork-on-gnu-linux> |

|  |
| --- |
| Problem: in streameye.c, it declare clients variable with NULL value. Later on, it can use clients[i] to access different client data    When there is new client coming in, streameye.c only use realloc() function    When there is client disconnected, it use realloc() function |
| Answer  From Open Groups' specifications (<https://pubs.opengroup.org/onlinepubs/009695399/functions/realloc.html>):  If ptr is a null pointer, realloc() shall be equivalent to malloc() for the specified size.  If ptr does not match a pointer returned earlier by calloc(), malloc(), or realloc() or if the space has previously been deallocated by a call to free() or realloc(), the behavior is undefined. |
| Reference:  <https://stackoverflow.com/questions/4459275/is-a-malloc-needed-before-a-realloc>  Dynamic allocate array: <https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/> |

|  |
| --- |
| Problem: What is sscanf function |
| Answer:  int sscanf(const char \*str, const char \*format, ...) reads formatted input from a string. |
| Reference:  <https://www.tutorialspoint.com/c_standard_library/c_function_sscanf.htm> |

|  |
| --- |
| Problem: How do I share variables between different .c files |
| Answer:  In fileA.c:   |  | | --- | | int myGlobal = 0; |   In fileA.h   |  | | --- | | **extern** int myGlobal; |   In fileB.c:   |  | | --- | | #include "fileA.h"  myGlobal = 1; |   So this is how it works:   * the variable lives in fileA.c * fileA.h tells the world that it exists, and what its type is (int) * fileB.c includes fileA.h so that the compiler knows about myGlobal before fileB.c tries to use it. |
| Reference:  <https://stackoverflow.com/questions/1045501/how-do-i-share-variables-between-different-c-files> |

|  |
| --- |
| Problem: How does makefile work |
| Answer: |
| Reference:  <https://opensource.com/article/18/8/what-how-makefile> |

|  |
| --- |
| Problem: What does it mean pointer plus/minus integer |
| Answer:  This question is about Pointer Arithmetic.  Pointers do not have to point to single variables. They can also point at the cells of an array. For example, we can write   |  | | --- | | int \*ip;  int a[10];  ip = &a[3]; |   and we would end up with ip pointing at the fourth cell of the array a (remember, arrays are 0-based, so a[0] is the first cell). We could illustrate the situation like this:    We'd use this ip just like the one in the previous section: \*ip gives us what ip points to, which in this case will be the value in a[3].  Once we have a pointer pointing into an array, we can start doing pointer arithmetic. Given that ip is a pointer to a[3], we can add 1 to ip:  ip + 1  What does it mean to add one to a pointer? In C, it gives a pointer to the cell one farther on, which in this case is a[4]. To make this clear, let's assign this new pointer to another pointer variable:  ip2 = ip + 1;  Now the picture looks like this:    If we now do  \*ip2 = 4;  we've set a[4] to 4. But it's not necessary to assign a new pointer value to a pointer variable in order to use it; we could also compute a new pointer value and use it immediately:  \*(ip + 1) = 5;  In this last example, we've changed a[4] again, setting it to 5. The parentheses are needed because the unary ``contents of'' operator \* has higher precedence (i.e., binds more tightly than) the addition operator. If we wrote \*ip + 1, without the parentheses, we'd be fetching the value pointed to by ip, and adding 1 to that value. The expression \*(ip + 1), on the other hand, accesses the value one past the one pointed to by ip.  Given that we can add 1 to a pointer, it's not surprising that we can add and subtract other numbers as well.  Of course, pointers are not limited to ints. It's quite common to use pointers to other types, especially char.  One question that comes up is whether the expression \*p++ increments p or what it points to. The answer is that it increments p. To increment what p points to, you can use (\*p)++.  When you're doing pointer arithmetic, you have to remember how big the array the pointer points into is, so that you don't ever point outside it.  Let’s see other code   |  | | --- | | #include<stdio.h>  #include<string.h>  #include<conio.h>  main()  {  char s[30], t[20];  char \*found;  /\* Entering the main string \*/  puts("Enter the first string: ");  gets(s);  /\* Entering the string whose position or index to be displayed \*/  puts("Enter the string to be searched: ");  gets(t);  /\*Searching string t in string s \*/  found=strstr(s,t);  if(found)  printf("Second String is found in the First String at %d position.\n", found - s);  else  printf("-1");  getch();  } |   Assuming you're wondering about the expression found-s, then what's happening is that you subtract two pointers.  Arrays naturally decay to pointers to their first element. That means plain s is equal to &s[0], which is what's happening here: found-s is equal to found - (&s[0]).  And the subtraction works because found is pointing to an element inside the array s, so the pointers are related (which is a requirement for pointer subtraction). The result is the difference (in elements) between the two pointers. |
| Reference:  <https://www.eskimo.com/~scs/cclass/notes/sx10b.html>  <https://stackoverflow.com/questions/60095585/how-can-a-character-array-be-subtracted-from-a-pointer> |

|  |
| --- |
| Problem: How to parse MJPEG file |
| Answer:  Since each JPEG starts with **0xFF 0xD8** as Start of Image marker and ends with **0xFF 0xD9**.  When processing multipart/x-mixed-replace, what you are *supposed* to do is:   1. read and discard the HTTP response body until you reach the first MIME boundary specified by the Content-Type response header. 2. then read a MIME entity's headers and data until you reach the next matching MIME boundary. 3. then process the entity's data as needed, according to its headers (for instance, displaying a image/jpeg entity onscreen). 4. if the connection has not been closed, and the last boundary read is not the termination boundary, go back to 2, otherwise stop processing the HTTP response. |
| Reference:  <https://stackoverflow.com/questions/47729941/mjpeg-over-http-specification> |

|  |
| --- |
| Problem: Want to use command line to execute python file.  1. Bash: syntax error near unexpected token ‘(‘ – Python  2. –bash: ./manage.py: Permision denied |
| Answer:  1. add #!/usr/bin/env python at the top of your script,  or call your script using **python myscript.py**  2. You need to make manage.py executable to excecute it.  Do chmod +x manage.py to make it excecutable. Alternately you can do python manage.py <cmd> instead. |
| Reference:  <https://stackoverflow.com/questions/10676050/bash-syntax-error-near-unexpected-token-python/10676069> |

|  |
| --- |
| Problem: |
| Answer: |
| Reference: |

|  |
| --- |
| Problem: |
| Answer: |
| Reference: |