{0}\_\_all\_\_ = 'BaseProactorEventLoop' NEWLINE  
  
{0}**import** io NEWLINE  
{0}**import** os NEWLINE  
{0}**import** socket NEWLINE  
{0}**import** warnings NEWLINE  
{0}**import** signal NEWLINE  
{0}**import** threading NEWLINE  
{0}**import** collections NEWLINE  
  
{0}**from** . **import** b ase\_events NEWLINE  
{0}**from** . **import** constants NEWLINE  
{0}**from** . **import** futures NEWLINE  
{0}**from** . **import** exceptions NEWLINE  
{0}**from** . **import** protocols NEWLINE  
{0}**from** . **import** sslproto NEWLINE  
{0}**from** . **import** transports NEWLINE  
{0}**from** . **import** trsock NEWLINE  
{0}**from** .log **import** logger NEWLINE  
  
  
{0}**def** *\_set\_socket\_extra*(transport, sock): NEWLINE  
 {4} INDENT transport*.\_extra*['socket'] = trsock.TransportSocket(sock) NEWLINE  
  
 {4}**try**: NEWLINE  
 {8} INDENT transport*.\_extra*['sockname'] = sock.getsockname() NEWLINE  
 {4} DEDENT **except** socket.error: NEWLINE  
 {8} INDENT **if** transport*.\_loop.get\_debug*(): NEWLINE  
 {12} INDENT logger.warning( NEWLINE  
 {16} INDENT "getsockname() failed on %r", sock, exc\_info=**True**) NEWLINE  
  
 {4} DEDENT DEDENT DEDENT **if** 'peername' **not** **in** transport*.\_extra*: NEWLINE  
 {8} INDENT **try**: NEWLINE  
 {12} INDENT transport*.\_extra*['peername'] = sock.getpeername() NEWLINE  
 {8} DEDENT **except** socket.error: NEWLINE  
 # UDP sockets may not have a peer name  
 {12} INDENT transport*.\_extra*['peername'] = **None** NEWLINE  
  
  
{0} DEDENT DEDENT DEDENT **class** *\_ProactorBasePipeTransport*(transports.\_FlowControlMixin,  
 transports.BaseTransport): NEWLINE  
 """Base class for pipe and socket transports."""  
  
 {4} INDENT **def** \_\_**in**it\_\_(**self**, loop, sock, protocol, waiter=**None**,  
 extra=**None**, server=**None**): NEWLINE  
 {8} INDENT super().\_\_init\_\_(extra, loop) NEWLINE  
 {8}**self***.\_set\_extra*(sock) NEWLINE  
 {8}**self***.\_sock* = sock NEWLINE  
 {8}**self***.set\_protocol*(protocol) NEWLINE  
 {8}**self***.\_server* = server NEWLINE  
 {8}**self***.\_buffer* = **None** NEWLINE # None or bytearray.  
 {8}**self***.\_read\_fut* = **None** NEWLINE  
 {8}**self***.\_write\_fut* = **None** NEWLINE  
 {8}**self***.\_pending\_write* = 0 NEWLINE  
 {8}**self***.\_conn\_lost* = 0 NEWLINE  
 {8}**self***.\_closing* = **False** NEWLINE # Set when close() called.  
 {8}**self***.\_eof\_written* = **False** NEWLINE  
 {8}**if** **self***.\_server* **is** **not** **None**: NEWLINE  
 {12}**self***.\_server.\_attach*() NEWLINE  
 {8}**self***.\_loop.call\_soon*(**self**.\_protocol.connection\_made, **self**) NEWLINE  
 {8}**if** waiter **is** **not** **None**: NEWLINE  
 # only wake up the waiter when connection\_made() has been called  
 {0} DEDENT DEDENT **self***.\_loop.call\_soon*(futures*.\_set\_result\_unless\_cancelled*,  
 waiter, **None**) NEWLINE  
  
 {4} INDENT **def** \_\_repr\_\_(**self**): NEWLINE  
 {8} INDENT info = [**self**.\_\_**class**\_\_.\_\_name\_\_] NEWLINE  
 {8}**if** **self***.\_sock* **is** **None**: NEWLINE  
 {12} INDENT info.append('closed') NEWLINE  
 {8} DEDENT **elif** **self***.\_clos****in****g*: NEWLINE  
 {12} INDENT info.append('closing') NEWLINE  
 {8} DEDENT **if** **self***.\_sock* **is** **not** **None**: NEWLINE  
 {12} INDENT info.append(f'fd={**self**.\_sock.fileno()}') NEWLINE  
 {8} DEDENT **if** **self***.\_read\_fut* **is** **not** **None**: NEWLINE  
 {12} INDENT info.append(f'read={**self**.\_read\_fut!r}') NEWLINE  
 {8} DEDENT **if** **self***.\_write\_fut* **is** **not** **None**: NEWLINE  
 {12} INDENT info.append(f'write={**self**.\_write\_fut!r}') NEWLINE   
 {8} DEDENT **if** **self***.\_buffer*: NEWLINE  
 {12} INDENT info.append(f'write\_bufsize={len(**self**.\_buffer)}') NEWLINE  
 {8} DEDENT **if** **self***.\_eof\_written*: NEWLINE  
 ? {12} INDENT info.append('EOF written') NEWLINE  
 {8} DEDENT **return** '<{}>'*.format*(' '.join(info)) NEWLINE   
  
 {4} DEDENT **def** *\_set\_extra*(**self**, sock): NEWLINE  
 {8} INDENT **self***.\_extra*['pipe'] = sock NEWLINE  
  
 {4} DEDENT **def** *set\_protocol*(**self**, protocol): NEWLINE  
 {8} INDENT **self***.\_protocol* = protocol NEWLINE  
  
 {4} DEDENT **def** *get\_protocol*(**self**): NEWLINE  
 {8} INDENT **return** **self***.\_protocol* NEWLINE  
  
 {4} DEDENT **def** **is**\_*closing*(**self**): NEWLINE  
 {8} INDENT **return** **self***.\_closing* NEWLINE  
  
 {4} DEDENT **def** *close*(**self**): NEWLINE  
 {8} INDENT **if** **self***.\_closing*: NEWLINE  
 {12} INDENT **return** NEWLINE  
 {8} DEDENT **self***.\_closing* = **True** NEWLINE  
 {8}**self***.\_conn\_lost* += 1 NEWLINE  
 {8}**if** **not** **self***.\_buffer* **and** **self***.\_write\_fut* **is** **None**: NEWLINE  
 {12} INDENT **self***.\_loop.call\_soon*(**self***.\_call\_connection\_lost*, **None**) NEWLINE  
 {8} DEDENT **if** **self***.\_read\_fut* **is** **not** **None**: NEWLINE  
 {12} INDENT **self***.\_read\_fut.cancel*() NEWLINE  
 {12}**self***.\_read\_fut* = **None** NEWLINE  
  
 {4} DEDENT DEDENT **def** \_\_**del**\_\_(**self**, *\_warn*=warnings.warn): NEWLINE  
 {8} INDENT **if** **self***.\_sock* **is** **not** **None**: NEWLINE  
 {12} INDENT *\_warn*(f"unclosed transport {**self**!r}", ResourceWarning, source=**self**) NEWLINE  
 {12}**self***.close*() NEWLINE  
  
 {4} DEDENT DEDENT **def** *\_fatal\_error*(**self**, exc, message='Fatal error on pipe transport'): NEWLINE  
 {8} INDENT **try**: NEWLINE  
 {12} INDENT **if** isinstance(exc, OSError): NEWLINE  
 {16} INDENT **if** **self***.\_loop.get\_debug*(): NEWLINE  
 {20} INDENT logger.debug("%r: %s", **self**, message, exc\_info=**True**) NEWLINE  
 {12} DEDENT DEDENT **else**: NEWLINE  
 {16} INDENT **self***.\_loop.call\_exception\_handler*({ NEWLINE  
 {20} INDENT 'message': message, NEWLINE  
 {20}'exception': exc, NEWLINE  
 {20}'transport': **self**, NEWLINE  
 {20}'protocol': **self***.\_protocol*, NEWLINE  
 })  
 {8} DEDENT DEDENT DEDENT **finally**: NEWLINE  
 {12} INDENT **self***.\_force\_close*(exc) NEWLINE  
  
 {4} DEDENT DEDENT **def** *\_force\_close*(**self**, exc): NEWLINE  
 {8} INDENT **if** **self***.\_empty\_waiter* **is** **not** **None** **and** **not** **self***.\_empty\_waiter.done*(): NEWLINE  
 {12} INDENT **if** exc **is** **None**: NEWLINE  
 {16} INDENT **self***.\_empty\_waiter.set\_result*(**None**) NEWLINE  
 {12} DEDENT **else**: NEWLINE  
 {16} INDENT **self***.\_empty\_waiter.set\_exception*(exc) NEWLINE  
 {8} DEDENT DEDENT **if** **self***.\_closing*: NEWLINE  
 {12} INDENT **return** NEWLINE  
 {8} DEDENT **self***.\_closing* = **True** NEWLINE  
 {8}**self***.\_conn\_lost* += 1 NEWLINE  
 {8}**if** **self***.\_write\_fut*: NEWLINE  
 {12} INDENT **self***.\_write\_fut.cancel*() NEWLINE  
 {12}**self***.\_write\_fut* = **None** NEWLINE  
 {8} DEDENT **if** **self***.\_read\_fut*: NEWLINE  
 {12} INDENT **self***.\_read\_fut.cancel*() NEWLINE  
 {12}**self***.\_read\_fut* = **None** NEWLINE  
 {8} DEDENT **self***.\_pending\_write* = 0 NEWLINE  
 {8}**self***.\_buffer* = **None** NEWLINE  
 {8}**self***.\_loop.call\_soon*(**self**.\_call\_connection\_lost, exc) NEWLINE