# Program 5

Yu yang 892449550

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.nio.ByteBuffer;

import java.nio.channels.AsynchronousFileChannel;

import java.nio.channels.CompletionHandler;

import java.nio.charset.Charset;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.logging.FileHandler;

import java.util.logging.Logger;

import java.util.logging.SimpleFormatter;

import static java.nio.file.StandardOpenOption.READ;

public class AsynchIO {

//input file name

private static String INPUT\_NAME;

private static final String EOLN = System.lineSeparator();

//java logger

private static Logger logger;

private static boolean finishFlag = false;

private static class ProgressInfo{

private int totalNumOfReq = 0;

private int processedNum = 0;

private boolean parseFinished = false;

}

//attachment

private class Attachment{

private Path path;

private int opid;

private ByteBuffer buf;

private AsynchronousFileChannel asyncChannel;

private ProgressInfo pInfo;

public Attachment(Path p, int opid, ByteBuffer bF, AsynchronousFileChannel asyncFC, ProgressInfo pI){

path = p;

this.opid = opid;

buf = bF;

asyncChannel = asyncFC;

pInfo = pI;

}

}

private class MyHandler implements CompletionHandler<Integer, Attachment> {

@Override

public void completed(Integer result, Attachment att){

//position out of bounds

if(result == -1){

logger.warning(String.format("op not performed because of start position out of file size bound. opid = %d, path = %s ",att.opid,att.path.toString()));

}else {

StringBuilder sB = new StringBuilder();

sB.append(att.opid + " ");

byte[] bufArr = att.buf.array();

//use ASCII

String outputChars = new String(att.buf.array(), Charset.forName("US-ASCII"));

sB.append(outputChars + EOLN);

System.out.print(sB.toString());

}

try{

att.asyncChannel.close();

att.pInfo.processedNum++;

if(att.pInfo.parseFinished&&(att.pInfo.processedNum==att.pInfo.totalNumOfReq)) {

System.out.println("DONE!!");

finishFlag = true;

System.exit(0);

}

}catch(IOException ex){

ex.printStackTrace();

System.exit(1);

}

}

@Override

public void failed(Throwable e, Attachment att){

logger.warning(String.format("op failed. opid = %d, path = %s",att.opid,att.path.toString()));

try{

att.asyncChannel.close();

att.pInfo.processedNum++;

if(att.pInfo.parseFinished&&(att.pInfo.processedNum==att.pInfo.totalNumOfReq)) {

System.out.println("DONE!!");

finishFlag = true;

System.exit(0);

}

}catch(IOException ex){

ex.printStackTrace();

}finally{

System.exit(1);

}

}

}

private void processOneCmd(int opid, String path, int offset, int cnt, ProgressInfo pI, Logger logger){

//log warning if parameter is invalid

if(offset < 0 || cnt < 0 || opid < 0){

pI.processedNum++;

logger.warning(String.format("Invalid commands: opid = %d, offset = %d, count = %d",opid,offset,cnt));

return;

}

Path filePath = Paths.get(path);

try {

AsynchronousFileChannel asyncFC = AsynchronousFileChannel.open(filePath, READ);

ByteBuffer bF = ByteBuffer.allocate(cnt);

MyHandler newHan = new MyHandler();

Attachment att = new Attachment(filePath,opid,bF,asyncFC,pI);

asyncFC.read(bF,offset,att,newHan);

}catch(IOException ex){

logger.warning(String.format("Invalid filePath: %s in opid %d",path,opid));

System.exit(1);

}

}

private void issueReadCommands(String fileName){

this.INPUT\_NAME = fileName;

AsynchIO.ProgressInfo pI = new AsynchIO.ProgressInfo();

try{

BufferedReader reader = new BufferedReader(new FileReader(INPUT\_NAME));

String request = null;

while((request = reader.readLine()) != null){

String para[] = request.split("\\s");

//process command

if(request.matches("\\d+\\s\\S+\\s\\d+\\s\\d+")){

pI.totalNumOfReq++;

processOneCmd(Integer.parseInt(para[0]),para[1],Integer.parseInt(para[2]),Integer.parseInt(para[3]),pI,logger);

} else {

logger.warning(String.format("Invalid commands: opid = %s, offset = %s, count = %s",para[0],para[1],para[2]));

}

}

//parse finish flag

pI.parseFinished = true;

reader.close();

}catch(FileNotFoundException ex){

logger.warning(String.format("Parsing input file failure. Input file name %s",INPUT\_NAME));

System.exit(1);

}catch(IOException ex){

ex.printStackTrace();

System.exit(1);

}finally{

if(pI.processedNum==pI.totalNumOfReq)

finishFlag = true;

}

}

public static void main(String[] args) throws Exception{

logger = Logger.getLogger(AsynchIO.class.getName());

FileHandler logFile;

try{

//use aio.log as log file

logFile = new FileHandler("aio.log");

logger.addHandler(logFile);

logFile.setFormatter(new SimpleFormatter());

//turn off the log info in console

logger.setUseParentHandlers(false);

} catch(Exception e){

e.printStackTrace();

}

new AsynchIO().issueReadCommands(args[0]);

//exit when finish flag is set

while(!finishFlag);

}

}