

# 보고서

## - 과제 2 -

과목명 | 응용소프트웨어실습

담당교수 | 이강훈

제출일 | 2019년 4월 28일

소속 | 소프트웨어학부

학번 | 2015202065

이름 | 윤홍찬



광운대학교  
KwangWoon University

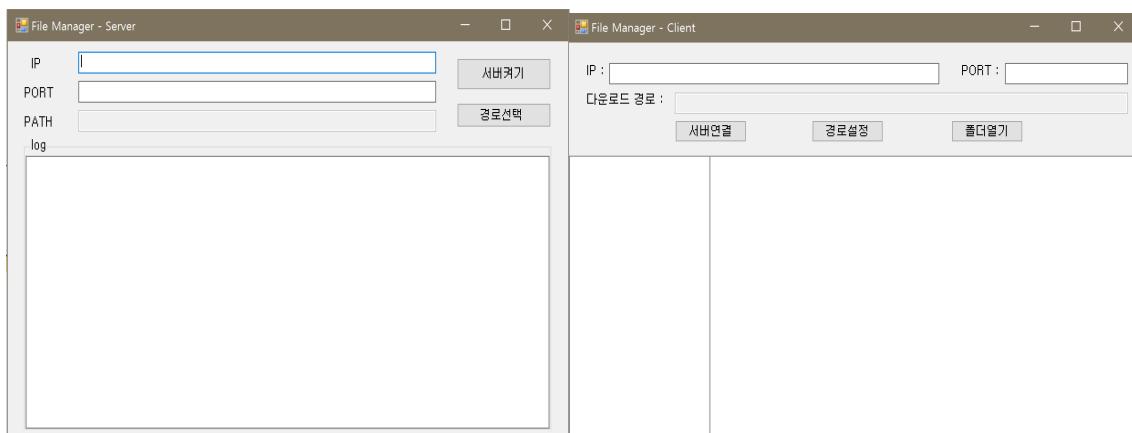
## < 목 차 >

I. 체크리스트 .....	1
II. Form 구성 .....	1
III. 서버, 클라이언트 연결 .....	2
IV. 원격 탐색기 기능 .....	4
V. 상세정보 .....	12
VI. 다운로드 .....	14
VII. 고찰 .....	18

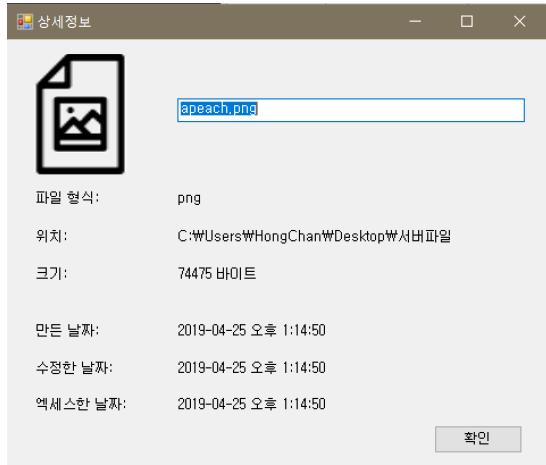
## I. 체크리스트

항목	구현내용	구현 여부	점수
Form 구성	Server Form 구성 (0.5점)	O	0.5
	Client Form 구성 (0.5점)	O	0.5
	Client Form2 구성 (0.5점)	O	0.5
서버, 클라이언트 연결	서버 IP, Port 번호 입력 (0.2점)	O	0.2
	서버 경로 설정 및 예외처리 (0.2점)	O	0.2
	클라이언트 IP, Port 번호 입력 (0.2점)	O	0.2
	클라이언트 경로 설정 및 예외처리 (0.2점)	O	0.2
	클라이언트 접속 시 서버측 접속 출력 - 소켓 접속 확인 (0.7점)	O	0.7
원격 탐색기 기능	ListView 및 TreeView 초기화 (0.5점)	O	0.5
	TreeView 확장 (0.5점)	O	0.5
	TreeView 선택 (1점)	O	1
	TreeView View 모드 (0.5점)	O	0.5
	ListView 더블 클릭 (1점)	O	1
상세정보 및 다운로드	우측 클릭 후 상세정보 클릭 시 상세정보 출력 (0.5점)	O	0.5
	더블 클릭 시 상세정보 출력 (0.5점)	O	0.5
	파일 전송 및 예외처리 (2.5점)	O	2.5
<b>총점</b>			<b>10</b>

## II. Form 구성



[ 왼쪽) Server Form / 오른쪽) Client Form ]

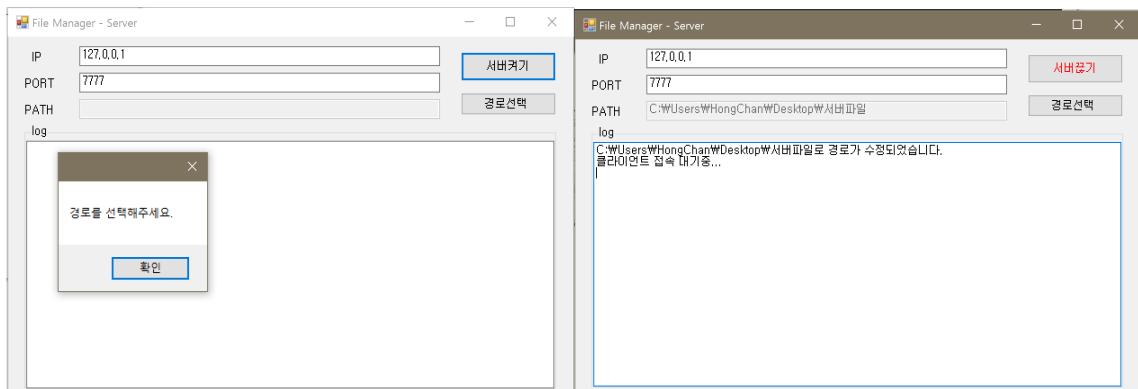


[ Client Form2 ]

윈도우 품을 구성하기 위해 강의 자료를 참고하였으며, 설명은 생략하겠습니다.

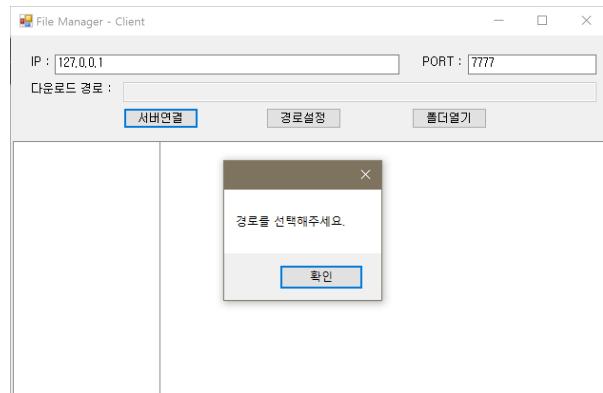
### III. 서버, 클라이언트 연결

#### 1. 서버

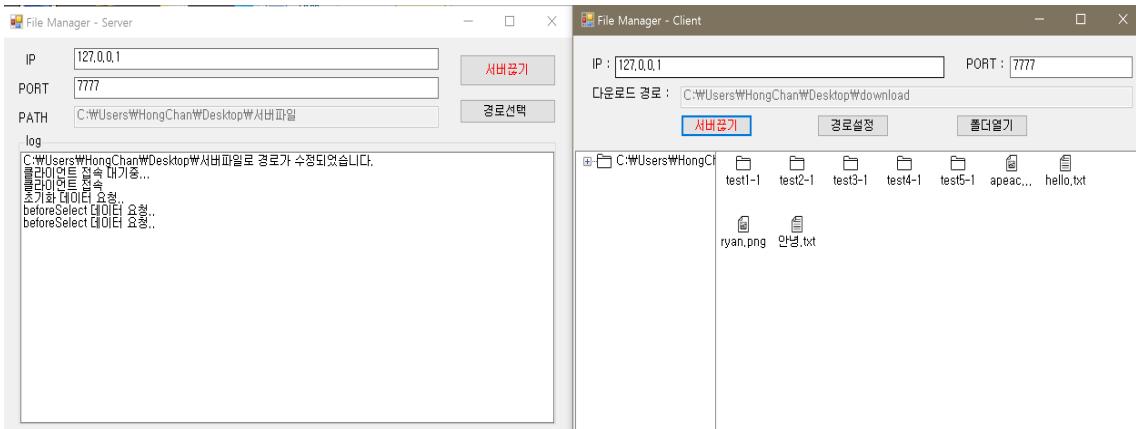


[ 왼쪽) Server - 경로 예외 처리 / 오른쪽) Server - 서버켜기 ]

#### 2. 클라이언트



[ Client - 경로 예외 처리 ]



[ Client 접속 시 Server측 접속 출력 - 소켓 접속 확인 ]

```

270     private void btnServer_Click(object sender, EventArgs e)
271     {
272         if (string.IsNullOrEmpty(txtPath.Text))
273         {
274             MessageBox.Show("경로를 선택해주세요.");
275         }
276         else
277         {
278             if(btnServer.Text == "서버켜기")
279             {
280                 m_thServer = new Thread(new ThreadStart(ServerStart));
281                 m_thServer.Start();
282
283                 btnServer.Text = "서버끊기";
284                 btnServer.ForeColor = Color.Red;
285             }
286             else
287             {
288                 ServerStop();
289                 btnServer.Text = "서버켜기";
290                 btnServer.ForeColor = Color.Black;
291             }
292         }
293     }

```

위의 코드는 Server측에서 ‘서버켜기’ 버튼을 눌렀을 때 발생하는 이벤트 핸들러 부분이다. 272행을 보면 경로 텍스트 박스에 값이 비어있으면(경로 설정이 되어있지 않으면) 경로를 선택해달라는 메시지를 출력하게 해 예외를 처리했다. 경로가 선택되어있으면 if문에 따라 서버 켜기/끄기에 맞는 코드가 실행된다.

```

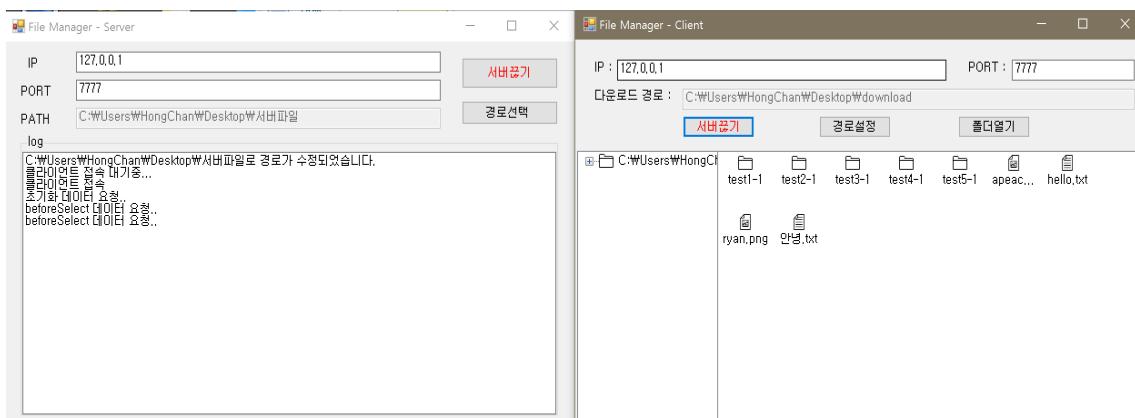
66     public void ServerStart()
67     {
68         try
69         {
70             string ip = txtIp.Text;
71             int port = int.Parse(txtPort.Text);
72             IPAddress ipAddr = IPAddress.Parse(ip);
73
74             m_server = new TcpListener(ipAddr, port);
75             m_server.Start();
76
77             m_bStop = true;
78             WriteLog("클라이언트 접속 대기중...");
```

위의 사진은 서버 켜기를 실행할 때 스레드에 의해 실행되는 코드이다. 이 부분은 강의 자료를 많이 참고하였는데, 75행에서 서버를 시작하고, ‘클라이언트 접속 대기’ 로그를 띠운 다음에 클라이언트가 연결되면 ‘클라이언트 접속’ 로그를 띠운 후 해당 클라이언트로부터 스트림을 얻는다.

클라이언트 코드의 경우 경로에 대한 예외처리는 맨 처음 코드와 비슷하게 작성했으며, 연결에 관한 부분은 이처럼 강의 자료를 참고하였으므로, 설명은 생략하겠다.

## M. 원격 탐색기 기능

### 1. 초기화



[ 클라이언트가 접속하면, 파일 및 디렉토리 정보를 가져와 클라이언트 TreeView와 ListView에 출력 + 파일 탑입에 맞는 아이콘 출력 ]

```

91     while (m_bConnect)
92     {
93         try
94         {
95             m_stream.Read(readBuffer, 0, 1024 * 4);
96         }
97         catch
98         {
99             WriteLog("서버에서 데이터를 읽는데 에러가 발생해 서버를 종료합니다.");
100            ServerStop();
101            this.Invoke(new MethodInvoker(delegate ()
102            {
103                btnServer.Text = "서버켜기";
104                btnServer.ForeColor = Color.Black;
105            }));
106            return;
107        }
108
109        Packet packet = (Packet)Packet.Desserialize(readBuffer);
110
111        switch ((int)packet.Type)

```

위의 코드는 **Server** 코드인데, 클라이언트가 연결된 후 연결이 지속되는 동안 계속 반복하는 반복문이다. 서버는 클라이언트로부터 온 패킷 데이터를 읽고 패킷 탑입을 비교해 해당하는 탑입에 맞는 코드가 실행되도록 switch문을 통해 구성하고 있다.

```
11     public enum PacketType
12     {
13         init = 0,
14         beforeSelect,
15         beforeExpand,
16         fileTransfer,
17         exitConnection
18     }
```

이는 위에서 언급한 패킷 타입의 종류이다. 차례대로 초기화 데이터 요청, ListView를 출력하기 위한 데이터 요청, TreeView를 출력하기 위한 데이터 요청, 파일 전송 요청, 클라이언트 연결 해제이다. 여기서는 초기화 패킷 타입에 대해 클라이언트가 요청하는 과정과 서버가 처리하는 것을 다루겠다. 각 패킷 타입들은 추후에 하나씩 설명할 것이다.

```
55     m_bConnect = true;
56     m_stream = m_client.GetStream();
57
58     /* Communicate with Server to Print TreeView and ListView*/
59     Init();
```

먼저 클라이언트의 코드이다. 클라이언트는 서버와 연결되자마자 Init()이라는 함수를 호출하여 ListView와 TreeView를 초기화 한다.

```
94     public void Init()
95     {
96         /* Send Init to Server */
97         Initialize init = new Initialize();
98         init.Type = (int)PacketType.init;
99         init.path = "";
100
101        Packet.Serialize(init).CopyTo(sendBuffer, 0);
102        Send();
103
104        /* Receive Path from Server */
105        int bytes = m_stream.Read(readBuffer, 0, 1024 * 4);
106        string path = Encoding.UTF8.GetString(readBuffer, 0, bytes);
107
108        /* Set TreeView */
109        TreeNode root;
110        root = trvDir.Nodes.Add(path);
111        root.ImageIndex = 1; // folder
112        if (trvDir.SelectedNode == null)
113            trvDir.SelectedNode = root;
114        root.SelectedImageIndex = root.ImageIndex;
115        root.Nodes.Add("");
116
117        /* Set ListView */
118        BeforeSelectMethod(path);
119    }
```

위에서 언급한 Init() 함수이다. 이 함수는 크게 4부분으로 나눌 수 있다. 처음 96행 ~ 102행은 서버가 선택한 경로를 얻기 위해 init 패킷 타입으로 설정해 서버에게 전송한다. 이후 104행 ~ 106행은 서버가 보낸 경로를 바이트로 얻어 이를 문자열로

바꾼다. 108행 ~ 115행은 해당 경로를 가지고 **TreeView**를 설정하는 과정이다. 여기서 111행은 폴더 사진으로 바꾸고 있으며, 115행은 ‘+’를 표시하기 위함이다. 마지막으로 118행은 **ListView**를 알맞게 출력하기 위해 함수를 호출하고 있는데, 이 함수의 내용은 바로 다음 3번 **TreeView** 선택에서 설명하겠다.

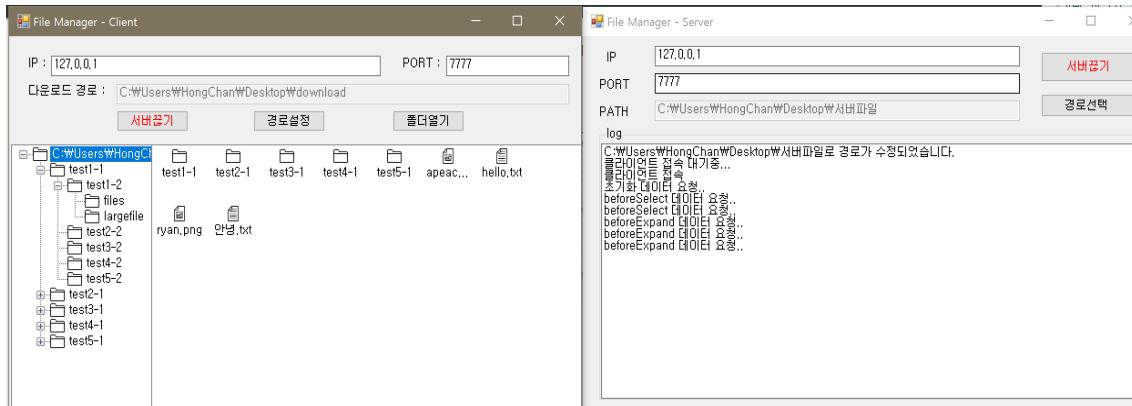
```

113     case (int)PacketType.init:
114     {
115         m_initializeClass = (Initialize)Packet.Deserialize(readBuffer);
116         WriteLog("초기화 데이터 요청...");
117
118         /* Send Path to client */
119         byte[] bytePath = Encoding.UTF8.GetBytes(dirPath);
120         m_stream.Write(bytePath, 0, bytePath.Length);
121         break;
122     }

```

이는 서버가 초기화 타입의 패킷을 처리하는 부분이다. 116행은 로그를 입력하고, 118행 ~ 120행까지는 클라이언트에게 서버에 설정된 경로를 보낸다.

## 2. **TreeView** 확장



[ **TreeView**의 확장버튼 클릭 시 필요한 정보를 서버에 요청해 출력 ]

```

76     [Serializable]
77     public class BeforeExpand : Packet
78     {
79         public string path = "";
80         public DirectoryInfo[] diArray;
81         public Dictionary<string, int> diAdd;
82     }

```

이는 **TreeView** 확장을 위해 클라이언트와 서버가 서로 주고받게 될 패킷 타입이다. 79행 path는 확장하기 위한 경로를 담고, diArray는 디렉토리 목록을 출력하기 위한 배열이고, 마지막으로 Dictionary 타입의 diAdd는 디렉토리를 표시할 때 ‘+’ 유무를 판별하기 위함이다.

```

207     private void trvDir_BeforeExpand(object sender, TreeViewCancelEventArgs e)
208     {
209         /* Send BeforeExpand to Server */
210         BeforeExpand be = new BeforeExpand();
211         be.Type = (int)PacketType.beforeExpand;
212         be.path = e.Node.FullPath;
213
214         Packet.Serialize(be).CopyTo(sendBuffer, 0);
215         Send();

```

이는 클라이언트의 TreeView BeforeExpand에 대한 이벤트 핸들러이다. 먼저 서버로부터 디렉토리 정보와 ‘+’ 표시 유무를 얻어오기 위해 beforeExpand 타입의 패킷에 경로를 담아 서버에 보낸다.

```

147     case (int)PacketType.beforeExpand:
148     {
149         m_beforeExpandClass = (BeforeExpand)Packet.Deserialize(readBuffer);
150         WriteLog("beforeExpand 데이터 요청..");
151         string path = m_beforeExpandClass.path;
152
153         /* Send Dir Array And Dictionary to client */
154         DirectoryInfo di;
155         DirectoryInfo[] diArrayPlus;
156         BeforeExpand be = new BeforeExpand();
157         try
158         {
159             di = new DirectoryInfo(path);
160             be.Type = (int)PacketType.beforeExpand;
161             be.diArray = di.GetDirectories();
162
163             /* To Set Plus */
164             be.diAdd = new Dictionary<string, int>();
165             foreach (DirectoryInfo dir in be.diArray)
166             {
167                 diPlus = new DirectoryInfo(dir.FullName);
168                 diArrayPlus = diPlus.GetDirectories();
169                 if (diArrayPlus.Length > 0)
170                     be.diAdd.Add(dir.Name, 1);
171                 else
172                     be.diAdd.Add(dir.Name, 0);
173             }
174             Packet.Serialize(be).CopyTo(sendBuffer, 0);
175             Send();
176

```

서버에서 beforeExpand 타입의 패킷을 처리하기 위한 코드이다. 150행에서 로그를 찍고, 160행 ~ 162행은 디렉토리 정보를 다시 패킷에 담았다. 165행 ~ 174행은 ‘+’ 유무를 확인하기 위한 코드이다. Dictionary타입을 사용해 디렉토리 이름을 Key 값으로 Value값이 0이면 하위 디렉토리가 없고, 1이면 하위 디렉토리가 있는 것이다. 마지막으로 175행 ~ 176행에서 다시 이 패킷을 클라이언트에게 전송한다.

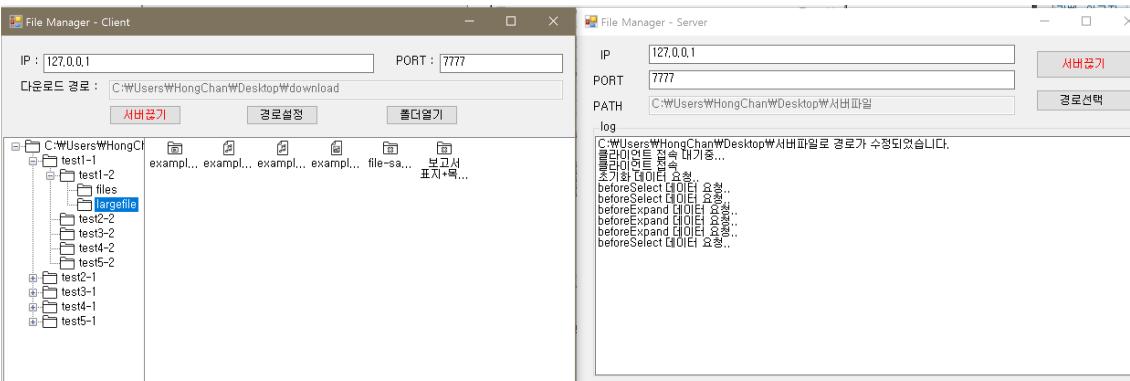
```

217     /* Receive Dir Array And Dictionary from Server */
218     m_stream.Read(readBuffer, 0, 1024 * 4);
219     BeforeExpand m_beforeExpandClass = (BeforeExpand)Packet.Deserialize(readBuffer);
220     Dictionary<string, int> dirDic = m_beforeExpandClass.diAdd;
221
222     /* Set TreeView */
223     TreeNode node;
224     e.Node.Nodes.Clear();
225     foreach (DirectoryInfo dirs in m_beforeExpandClass.diArray)
226     {
227         node = e.Node.Nodes.Add(dirs.Name);
228         node.ImageIndex = 1;
229
230         /* Set Plus */
231         if(dirDic[dirs.Name] == 1)
232         {
233             node.Nodes.Add(" ");
234         }
235     }

```

다시 클라이언트 이벤트 핸들러 코드이다. 위에서 보았듯이 서버에게 패킷을 전송하고 나서 217행 ~ 220행은 서버로부터 디렉토리 정보와 하위 디렉토리 유무의 정보를 받고, 222행 ~ 235행은 TreeView를 설정하면서 ‘+’ 유무 또한 설정한다.

### 3. TreeView 선택 및 ListView 이미지 설정



[TreeView의 메뉴 선택 시 해당 폴더의 파일 및 폴더를 서버에 요청해 ListView에 출력 ]

```

68     [Serializable]
69     public class BeforeSelect : Packet
70     {
71         public string path = "";
72         public DirectoryInfo[] diArray;
73         public FileInfo[] fiArray;
74     }

```

이는 TreeView 선택을 위해 클라이언트와 서버가 서로 주고받게 될 패킷 타입이다. 71행 path는 해당 디렉토리 경로를 담고, 각각 디렉토리와 파일에 대한 정보를 담는다.

```

238     private void trvDir_BeforeSelect(object sender, TreeViewCancelEventArgs e)
239     {
240         BeforeSelectMethod(e.Node.FullPath);
241     }

```

이는 클라이언트의 TreeView BeforeSelect에 대한 이벤트 핸들러이다. 1번 초기화 과정에서도 언급한 함수를 경로와 함께 호출하게 된다.

```

121     public void BeforeSelectMethod(string dirPath)
122     {
123         /* Send BeforeSelect to Server */
124         BeforeSelect bs = new BeforeSelect();
125         bs.Type = (int)PacketType.beforeSelect;
126         bs.path = dirPath;
127
128         Packet.Serialize(bs).CopyTo(sendBuffer, 0);
129         Send();

```

바로 위에서 호출한 함수이다. 먼저 서버로부터 디렉토리 정보와 파일 정보를 얻어오기 위해 beforeSelect 타입의 패킷에 경로를 담아 서버에 보낸다.

```

123     case (int)PacketType.beforeSelect:
124     {
125         m_beforeSelectClass = (BeforeSelect)Packet.Deserialize(readBuffer);
126         WriteLog("beforeSelect 데이터 요청..");
127         string path = m_beforeSelectClass.path;
128
129         /* Send Dir and File Array to client */
130         DirectoryInfo di;
131         BeforeSelect bs = new BeforeSelect();
132         try
133         {
134             di = new DirectoryInfo(path);
135             bs.Type = (int)PacketType.beforeSelect;
136             bs.diArray = di.GetDirectories();
137             bs.fiArray = di.GetFiles();
138             Packet.Serialize(bs).CopyTo(sendBuffer, 0);
139         }
140     }

```

서버에서 beforeSelect 타입의 패킷을 처리하기 위한 코드이다. 126행에서 로그를 찍고, 130행 ~ 140행은 해당 경로의 디렉토리 정보와 파일 정보를 패킷에 담아 클라이언트에게 다시 전송한다.

```

131     /* Receive Dir and File Array from Server */
132     m_stream.Read(readBuffer, 0, 1024 * 4);
133     BeforeSelect m_beforeSelectClass = (BeforeSelect)Packet.Deserialize(readBuffer);
134
135     /* Set ListView */
136     ListViewItem item;
137     lvwDir.Items.Clear();
138     foreach (DirectoryInfo tdis in m_beforeSelectClass.diArray)
139     {
140         item = lvwDir.Items.Add(tdis.Name);
141         item.SubItems.Add("");
142         item.SubItems.Add(tdis.LastWriteTime.ToString());
143         item.ImageIndex = 1;
144         item.Tag = "D";
145     }

```

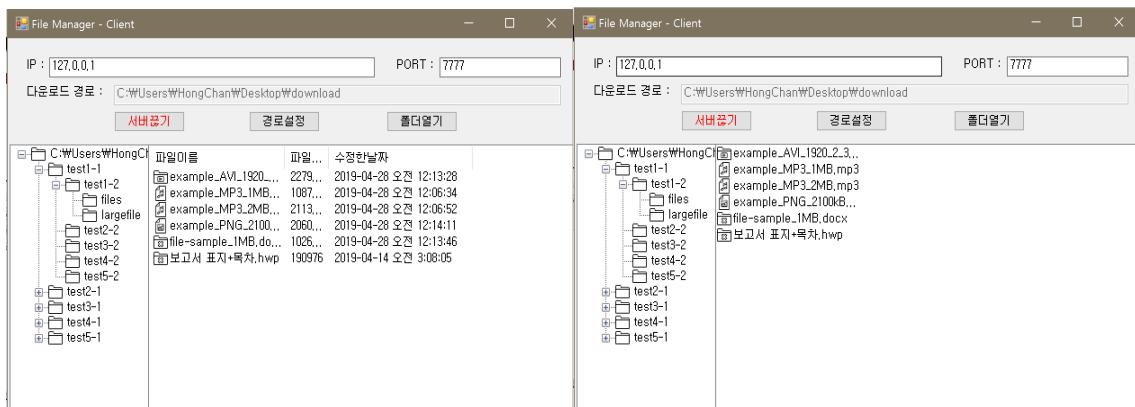
다시 클라이언트의 코드이다. 위에서 보았듯이 서버에게 패킷을 전송하고 나서 131행 ~ 133행은 서버로부터 디렉토리와 파일의 정보를 담고 있는 패킷을 받고, 135행 ~ 145행은 서버로부터 받은 디렉토리 정보를 가지고 ListView에 디렉토리를 추가한다. 143행은 디렉토리 이미지로 설정하기 위한 것이다.

```

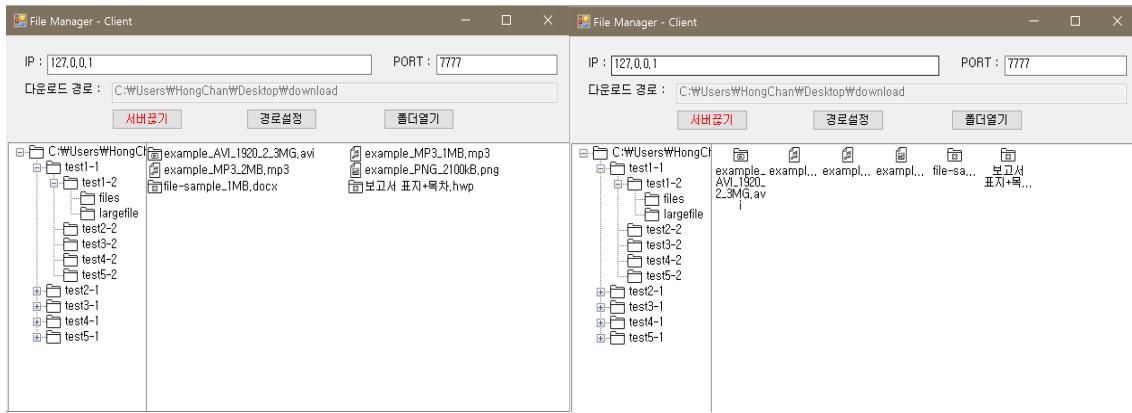
147     fileInfoDic.Clear();
148     foreach (FileInfo fis in m_beforeSelectClass.fiArray)
149     {
150         /* To Use File Detail */
151         if (!fileInfoDic.ContainsKey(fis.Name))
152             fileInfoDic.Add(fis.Name, fis);
153
154         item = lvwDir.Items.Add(fis.Name);
155         item.SubItems.Add(fis.Length.ToString());
156         item.SubItems.Add(fis.LastWriteTime.ToString());
157         if (fis.Extension == ".avi" || fis.Extension == ".wmv" || fis.Extension == ".mpg")
158             item.ImageIndex = 0;
159         else if (fis.Extension == ".jpg" || fis.Extension == ".png" || fis.Extension == ".jpeg")
160             item.ImageIndex = 2;
161         else if (fis.Extension == ".mp3" || fis.Extension == ".wav")
162             item.ImageIndex = 3;
163         else if (fis.Extension == ".txt")
164             item.ImageIndex = 5;
165         else
166             item.ImageIndex = 4;
167         item.Tag = "F";
168     }
169 }
```

이후 147행 ~ 169행은 ListView에 파일을 추가한다. 이때 157행 ~ 166행은 if문을 사용해 파일의 확장자별로 다른 이미지 파일을 적용하였다.

#### 4. TreeView View 모드



[ 왼쪽) 자세히 / 오른쪽) 간단히 ]



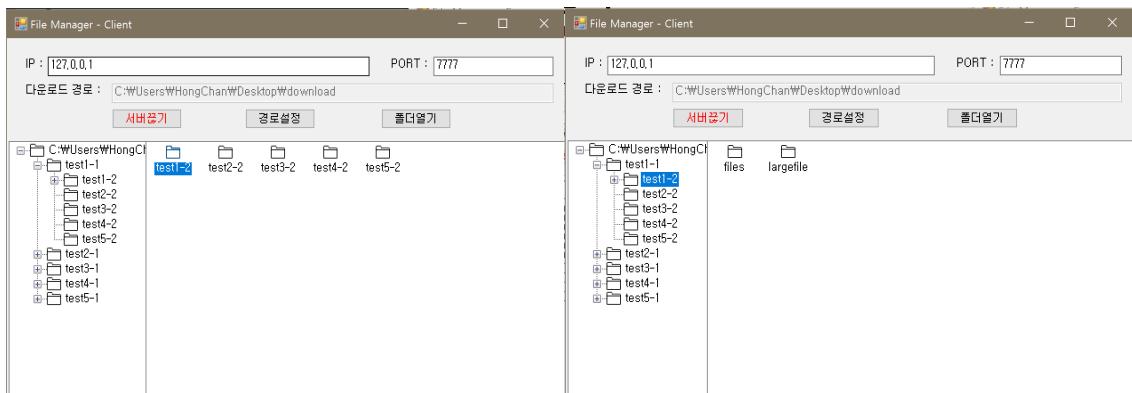
[ 왼쪽) 작은 아이콘 / 오른쪽) 큰 아이콘 ]

```

252     switch (item.Text)
253     {
254         case "자세히":
255             mnuDetail.Checked = true;
256             lvwDir.View = View.Details;
257             break;
258         case "간단히":
259             mnuList.Checked = true;
260             lvwDir.View = View.List;
261             break;
262         case "작은 아이콘":
263             mnuSmall.Checked = true;
264             lvwDir.View = View.SmallIcon;
265             break;
266         case "큰 아이콘":
267             mnuLarge.Checked = true;
268             lvwDir.View = View.LargeIcon;
269             break;
270     }
  
```

위와 같이 이벤트 핸들러를 등록하여 구현하였다. 강의자료를 참고하였기 때문에 자세한 설명은 생략한다.

## 5. ListView 더블클릭



[ ListView의 폴더를 더블 클릭 시 폴더로 이동 후 파일 및 폴더 정보를 출력 ]

```

307     private void lvwDir_DoubleClick(object sender, EventArgs e)
308     {
309         ListView.SelectedListViewItemCollection siList;
310         siList = lvwDir.SelectedItems;
311
312         foreach (ListViewItem item in siList)
313         {
314             OpenItem(item);
315         }
316     }

```

이는 클라이언트의 ListView DoubleClick에 대한 이벤트 핸들러이다. ListView에서 클릭한 요소에 대해 OpenItem() 함수를 호출한다.

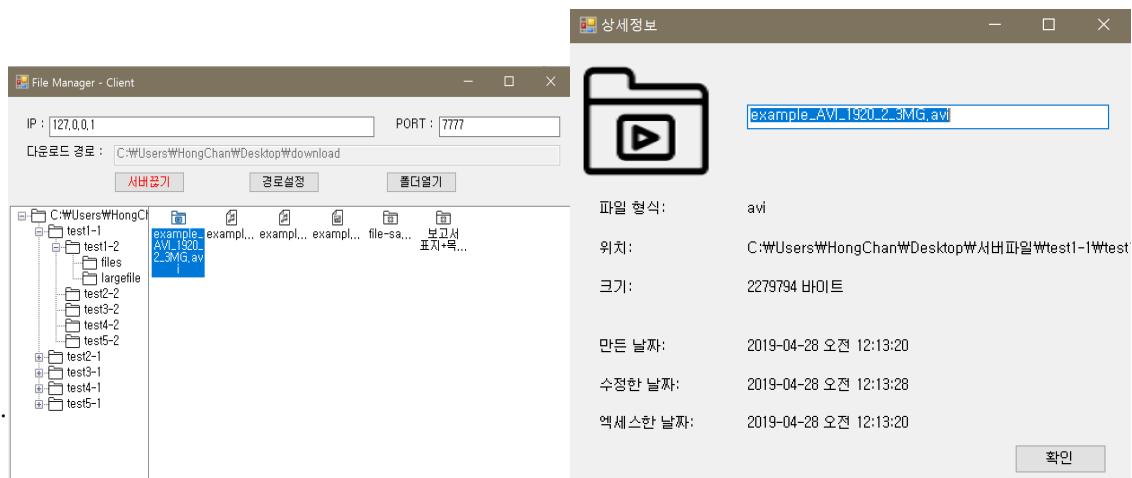
```

273     public void OpenItem(ListViewItem item)
274     {
275         TreeNode node;
276         TreeNode child;
277
278         if(item.Tag.ToString() == "D")
279         {
280             node = trvDir.SelectedNode;
281             /* BeforeExpand Event */
282             node.Expand();
283
284             child = node.FirstNode;
285
286             while(child != null)
287             {
288                 if(child.Text == item.Text)
289                 {
290                     /* BeforeSelect Event */
291                     trvDir.SelectedNode = child;
292                     trvDir.Focus();
293                 }
294                 child = child.NextNode;
295             }
296         }

```

위에서 호출한 함수이다. 더블클릭 한 요소가 디렉토리라면, 먼저 해당 TreeView를 Expand한다.(이때, BeforeExpand 이벤트가 발생) 이후 그의 자식 노드들을 순회하며 클릭한 것과 동일한 노드를 찾아 Select한다.(이때, BeforeSelect 이벤트 발생)

## V. 상세정보



[ 클라이언트의 파일의 우클릭 후 상세정보 클릭 시 위와 같이 상세정보 출력  
+ 클라이언트의 파일을 더블 클릭 시 위와 같이 상세정보 출력 ]

```
363     |     private void mnMore_Click(object sender, EventArgs e)
364     |     {
365     |     |     openFileDialog = new FormDetail();
366     |     |     bool isTrue = SetDetail();
367     |     |     if(isTrue)
368     |     |     openFileDialog.Show();
369     |     }
```

이는 클라이언트의 상세보기 Click에 대한 이벤트 핸들러이다. Client form2인 openFileDialog의 객체를 새로 생성하고, 폼의 내용을 채우기 위해 SetDetail() 함수를 호출한다.

```
297     |     else if(item.Tag.ToString() == "F")
298     |     {
299     |     |     // Already Clicked Change
300     |     |     openFileDialog = new FormDetail();
301     |     |     bool isTrue = SetDetail();
302     |     |     if (isTrue)
303     |     |     openFileDialog.Show();
304     |     }
```

이는 위에서 살펴본 클라이언트의 ListView DoubleClick에 대한 처리 부분이다. 선택한 요소가 파일이라면, 위와 똑같이 객체를 새로 생성하고 폼의 내용을 채우기 위해 SetDetail() 함수를 호출한다. 만약 선택한 요소가 파일이라면 SetDetail() 함수 호출로 인해 안의 요소의 값들이 채워지고 Dialog를 Show 하게 된다.

```
323     |     public bool SetDetail()
324     |     {
325     |     |     ListView.SelectedListViewItemCollection siList = lvwDir.SelectedItems;
326     |     |     ListViewItem lvwItem = siList[0];
327     |
328     |     |     if (lvwItem.Tag.ToString() == "D")
329     |     |     {
330     |     |     MessageBox.Show("폴더는 상세정보를 지원하지 않습니다.");
331     |     |     return false;
332     |     }
```

위에서 호출한 함수이다. 만약 선택한 요소가 디렉토리라면, 메시지를 출력하고 false를 return해 Client form2 Dialog가 뜨지 않게 된다.

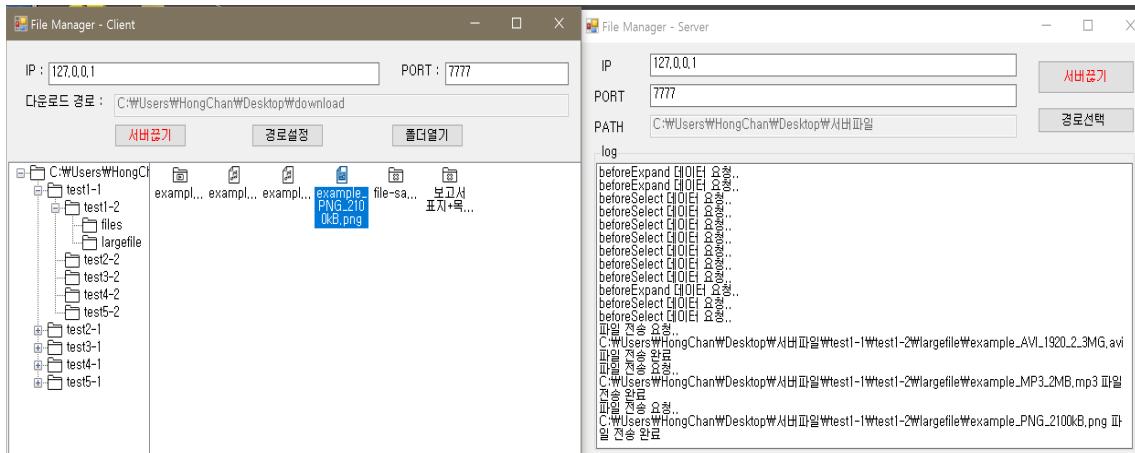
```

333     {
334         FileInfo fi = fileInfoDic[lvwItem.SubItems[0].Text];
335
336         /* Change Labels And Image */
337         fileDialog.txtName.Text = fi.Name;
338         fileDialog.picBox.SizeMode = PictureBoxSizeMode.StretchImage;
339         if (fi.Extension == ".avi" || fi.Extension == ".wmv" || fi.Extension == ".mpg")
340             fileDialog.picBox.Image = DetailImgList.Images[0];
341         else if (fi.Extension == ".jpg" || fi.Extension == ".png" || fi.Extension == ".jpeg")
342             fileDialog.picBox.Image = DetailImgList.Images[2];
343         else if (fi.Extension == ".mp3" || fi.Extension == ".wav")
344             fileDialog.picBox.Image = DetailImgList.Images[3];
345         else if (fi.Extension == ".txt")
346             fileDialog.picBox.Image = DetailImgList.Images[5];
347         else
348             fileDialog.picBox.Image = DetailImgList.Images[4];
349
350         fileDialog.lblFormat.Text = fi.Extension.Substring(1); /* Erase '.' */
351         fileDialog.lblLocation.Text = fi.DirectoryName;
352         fileDialog.lblSize.Text = fi.Length.ToString() + " 바이트";
353
354         fileDialog.lblMade.Text = fi.CreationTime.ToString();
355         fileDialog.lblChange.Text = fi.LastWriteTime.ToString();
356         fileDialog.lblAccess.Text = fi.LastAccessTime.ToString();
357
358         return true;
359     }

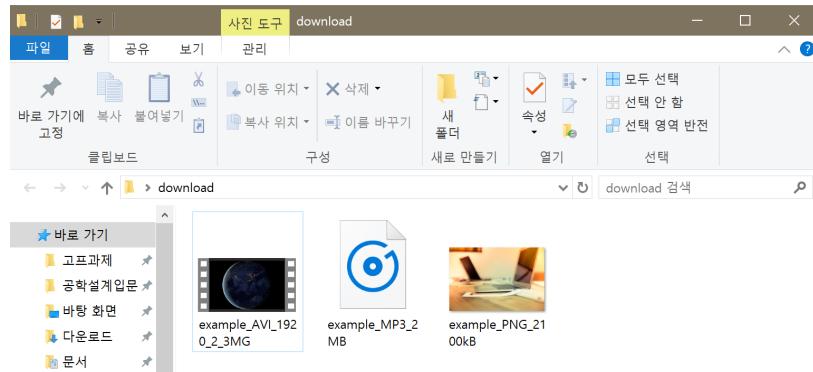
```

335행에서 `fileInfoDic`이라는 `Dictionary` 타입으로부터 `FileInfo` 객체를 얻어내고 있는데, 이 `fileInfoDic`은 `SelectBefore` 이벤트가 발생해 `ListView`에 표시되는 게 달라질 때마다 파일 이름을 Key 값으로, 파일의 정보를 담는 `FileInfo`를 Value으로 저장한다. 이후, 다시 `SelectBefore` 이벤트가 발생해 `ListView`에 표시되는 게 달라지면, `fileInfoDic`에 모든 요소를 지우고 다시 새로운 값으로 채우게 된다. 이렇게 `fileInfoDic`은 최소한의 공간을 사용해 최신 파일의 정보를 담고 있다. 338행 ~ 357행은 이렇게 얻은 파일의 정보로 사진, 텍스트박스, 각 라벨을 설정한다.

## VI . 다운로드



[ 클라이언트의 파일의 우클릭 후 다운로드 클릭 시 다운로드 경로로 파일을 다운로드 ]



[ 성공적으로 다운로드 된 모습 (2MB인 파일을 대상으로 함) ]

파일명	파일 형식	크기	만든 날짜	수정한 날짜	액세스한 날짜
example_AVI_192_0_2_3MG.avi	avi	2.17MB (2,279,794 바이트)	2019년 4월 28일	2019년 4월 28일	2019년 4월 28일
example_MP3_2MB.mp3	mp3	2.01MB (2,113,939 바이트)	2019년 4월 28일	2019년 4월 28일	2019년 4월 28일
example_PNG_2100kB.png	png	1.96MB (2,064,384 바이트)	2019년 4월 28일	2019년 4월 28일	2019년 4월 28일

[ 다운로드 받은 파일의 속성과 원본 파일의 상세정보가 같음을 확인 ]

```

84     [Serializable]
85     public class FileTransfer : Packet
86     {
87         public string path;
88         public long size;
89     }

```

이는 파일 전송을 위해 클라이언트와 서버가 서로 주고받게 될 패킷 타입이다. 87행 path는 해당 파일의 경로를 담고, 88행은 해당 파일의 크기를 담는다.

```

371     private void mnuDownload_Click(object sender, EventArgs e)
372     {
373         ListView.SelectedListViewItemCollection siList = lvwDir.SelectedItems;
374         ListViewItem lwlItem = siList[0];
375
376         if (lwlItem.Tag.ToString() == "D")
377         {
378             MessageBox.Show("폴더는 다운로드를 지원하지 않습니다.");
379         }
380         else
381         {
382             FileInfo fi = fileInfoDic[lwlItem.SubItems[0].Text];
383             //long currentLength = 0;
384             long totalLength = fi.Length;
385             byte[] fileRecvBuffer = new byte[1024];
386
387             /* Send File Path To Server */
388             FileTransfer ft = new FileTransfer();
389             ft.Type = (int)PacketType.fileTransfer;
390             ft.path = fi.DirectoryName + "\\\" + lwlItem.SubItems[0].Text;
391             ft.size = totalLength;
392
393             Packet.Serialize(ft).CopyTo(sendBuffer, 0);
394             Send();
395         }
396     }

```

이는 클라이언트의 다운로드 Click에 대한 이벤트 핸들러이다. 376행 ~ 379행은 선택한 요소가 디렉토리라면, 예외 메시지를 출력하게 된다. 382행 ~ 385행은 위에서 언급한 것과 마찬가지로 선택한 요소 파일의 정보인 FileInfo 객체를 통해 다운로드 받을 파일의 크기를 저장한다. 이후 388행 ~ 394행은 패킷에 다운로드 받을 파일의 경로와 크기를 담아 서버에 전송한다.

```

396     /* Create File And FileStream */
397     string path = txtPath.Text + "\\\" + lwlItem.SubItems[0].Text;
398     FileStream fStr = new FileStream(path, FileMode.Create, FileAccess.Write);
399     BinaryWriter bWriter = new BinaryWriter(fStr);
400     long loopCnt = (long)(totalLength / 1024 + 1);
401
402     /* Download File */
403     int reSize = 1024;
404     //int sum = 0;
405     for (long i = 0; i < loopCnt; i++)
406     {
407         if (i == loopCnt - 1)
408             reSize = (int)(totalLength - (1024 * (loopCnt - 1)));
409
410         m_stream.Read(fileRecvBuffer, 0, reSize);
411         bWriter.Write(fileRecvBuffer, 0, reSize);
412         //sum += reSize;
413     }
414     fStr.Close();
415     bWriter.Close();
416 }

```

이후 397행 ~ 398행은 FileStream을 이용하여 다운로드 경로에 파일을 생성하고, 쓰기상태로 연다. 399행은 파일에 데이터를 쓰기 위한 BinaryWriter 객체를 생성하였고, 400행은 파일의 크기를 배열의 크기로 나누어 반복문이 몇 번 반복되어야 하는지 계산한다. 이후 403행 ~ 413행은 파일 데이터를 서버로부터 받는 부분이다. 414행 ~ 415행은 이용한 스트림을 닫는다.

```
184     case (int)PacketType.fileTransfer:
185     {
186         m_fileTransferClass = (FileTransfer)Packet.Deserialize(readBuffer);
187         WriteLog("파일 전송 요청..");
188         string path = m_fileTransferClass.path;
189         long size = m_fileTransferClass.size;
190         byte[] fileSendBuffer = new byte[1024];
191
192         /* Create File Stream */
193         FileStream fStr = new FileStream(path, FileMode.Open, FileAccess.Read);
194         BinaryReader bReader = new BinaryReader(fStr);
195         long loopCnt = (long)(size / 1024 + 1);
```

서버에서 FileTransfer 타입의 패킷을 처리하기 위한 코드이다. 186행 ~ 190행은 클라이언트로부터 파일을 전송할 경로와 파일의 크기를 받고 193행 ~ 195행은 해당하는 경로에 있는 파일을 읽기 전용으로 열고 전체 반복문이 몇 번 반복해야 하는지 계산한다.

```
197     try
198     {
199         /* Send File */
200         int reSize = 1024;
201         for (long i = 0; i < loopCnt; i++)
202         {
203             if (i == loopCnt - 1)
204                 reSize = (int)(size - (1024 * (loopCnt - 1)));
205
206             fileSendBuffer = bReader.ReadBytes(reSize);
207             m_stream.Write(fileSendBuffer, 0, reSize);
208             m_stream.Flush();
209
210             /* Reset Array */
211             for (int j = 0; j < reSize; j++)
212             {
213                 fileSendBuffer[j] = 0;
214             }
215
216             WriteLog(path + " 파일 전송 완료");
217     }
```

이후 197행 ~ 217행은 반복문을 돌면서 파일로부터 데이터를 읽어 들이며 클라이언트로 파일 데이터를 전송하고, 파일 전송이 완료되면 로그를 찍는다.

## VII. 고찰

2학년 2학기 시스템소프트웨어실습 강의에서 마지막 프로젝트로 소켓 프로그래밍을 경험해보았다. 이어 이번 2차 과제에서도 소켓 프로그래밍을 이용한 파일 탐색기 및 다운로드를 구현하여, 소켓을 이용한 서버와 클라이언트 통신에 대한 부분은 이제 많이 익숙해져 숙달이 된 것 같다. 이번 2차 과제를 진행하면서 어려움이 있었던 부분은 파일 다운로드를 구현하는 부분이었다. 처음에 파일이 아예 전송되지 않거나, 파일 데이터가 달라지는 등의 문제가 있었지만, 꾸준히 시간을 투자해 문제를 해결하고 완벽히 과제를 구현하였다. 이번 응용소프트웨어실습 팀 프로젝트로 소켓 통신을 이용한 서버-클라이언트 게임을 개발하기로 하였는데, 과제로 숙달된 소켓 프로그래밍 실력으로 완벽하게 팀 프로젝트 구현에 임해야겠다고 생각했다.