Sesi/Perkuliahan ke: I

Tujuan Instruksional Khusus :

1. Mahasiswa dapat menjelaskan pengertian tentang suatu file.
2. Mahasiswa dapat menyebutkan procedure dan fungsi standart untuk semua tipe file.
3. Mahasiswa dapat membuat deklarasi untuk suatu file.
4. Mahasiswa mengerti cara membuat file, menambah data dan menampilkan data pada file
5. Mahasiswa dapat membuat contoh program.

Pokok Bahasan : FILE (BERKAS)

Deskripsi singkat : Dalam pertemuan ini akan mempelajari tentang Jenis File & Operasinya, Pendeklarasian pada file, Procedure dan fungsi standart untuk semua tipe file, Menangani Kesalahan File (I/O) Error, Membuat file text, Menambah data, Menampilkan semua data

Referensi :

1. Anonim, *"Algoritma & Pemrograman II"*, Penerbit Gunadarma, Jakarta, 1990

2. Bryon, Goffried, *"Programming with PASCAL"*, Schaum Series, New York, 1986

3. Prather, Ronald E., *"Problem Solving Principle : Programming with Pascal"*, Prentice Hall, New Jersey, 1982

4. Dumairy, *Matematika Terapan untuk Bisnis & Ekonomi* Press (BPFE Yogyakarta)

5. Yogianto H.M, *Turbo Pascal 5.0 Penerbit Andi Offset*, Yogyakarta.

6. Ediman Lukito*, Belajar Sendiri Pemrograman dengan Turbo Pascal 7.0*

7. Ir. P. Insap Santosa, *Turbo Pascal versi 5.0 dan 5.5*, Elexmedia Komputindo

FILE (BERKAS)

File adalah kumpulan byte-byte yang disimpan dalam media penyimpanan. Merupakan komponen yang bertipe data sama, yang jumlahnya tidak tentu, yang dapat ditambah atau dikurangi jika dibutuhkan.

Pointer file adalah bagian yang menunjuk ke komponen file yang akan diakses (dibaca atau direkam) untuk keperluan pengaksesan file (akan dibahas kemudian).

File pada Pascal dikenal dalam 3 jenis, yaitu :

1. File Text
2. File bertipe
3. File tidak bertipe

Selain kita akan mempelajari tentang bagaimana membuat sebuah file atau menambahkan isi suatu file, kita dapat pula melakukan manipulasi File, yaitu :

1. Menggunakan parameter
2. Mengenai Atribut File
3. Menghapus file
4. Mengubah nama file
5. Mengenai tanggal dan waktu file
6. Mencari file
7. Mengecek keberadaan file
8. Memberikan directory file

1. File Text

1.1. Karakteristik

- Berisi data karakter ASCII
- Tiap record boleh memiliki panjang yang bervariasi
- Setiap record diakhiri tanda EOL (end of Line)
- Hanya dapat diakses secara sequensial (berurutan).
- Isi filenya dapat dilihat oleh perintah dos type atau editor text

1.2. Membuat file text

Urutan Prosesnya :

1. Mendeklarasikan variabel file

   ```
   Var NmVar:TEXT;
   ```

   Dengan :

   NmVar : Nama variabel file text

2. Menghubungkan variabel file dengan nama file

   ```
   Assign (NmVar, nama file);
   ```

   Dengan :

   NmVar : Nama variabel file text

   nama file : Nama file dalam bentuk string, format 8:3 penamaan dos, ditulis dalam bentuk string.

3. Membuat file text aktif

   ```
   Rewrite(NmVar);
   ```

   Dengan :

   NmVar : Nama variabel file text yang sudah di- assign

4. Menulis ke dalam file text

   Write / writeln (NmVar, data item1, data item 2, …)

   Dengan :

   NmVar : Nama variabel file text

   Data item : text / string yang akan dituliskan, atau bisa juga berupa isi suatu variabel

5. Menutup file

   Close (NmVar);

   Dengan:

   NmVar : Nama variabel file text

Contoh :

```
Program membuat_file_text_namafile_HALLO_TXT;
Uses crt;
Var var_teks:TEXT;
Begin
    Clrscr;
    Assign(var_teks,'Hallo.txt');
    Rewrite(var_teks);
    Writeln(var_teks,'Hallo… ini program pertamaku!');
    Writeln(var_teks,'Contoh file teks');
    Writeln(var_teks,'---------------------------------------');
    Close(var_teks);
End.
```

1.3. Membaca File Text

Urutan Prosesnya :

1. Mendeklarasikan variabel file

   ```
   Var NmVar:TEXT;
   ```

2. Menghubungkan variabel file dengan nama file

   ```
   Assign (NmVar, nama file)
   ```

3. Membaca isi file dan menampilkannya di layar

   ```
   While not eof (NmVar) do
   Begin
   Read / readln ( NmVar, data item 1, data item 2, … );
   Write / writeln ( data item1, data item 2, … );
   End;
   ```

4. Menutup file

   ```
   Close (NmVar);
   ```

Contoh :

```
Program membaca_file_HALLO_TXT;
Uses crt;
Var var_teks:TEXT;
     Pesan:string;
Begin
    Clrscr;
```

```
    Assign(var_teks,'Hallo.txt');
    Reset(var_teks);
    While not eof (var_teks) do
        Begin
            Readln ( var_teks, pesan );
            Writeln (pesan);
        End;
    Close(var_teks);
End.
```

1.4. Menambah isi File Text

Urutan Prosesnya :

1. Mendeklarasikan variabel file

   ```
   Var NmVar:TEXT;
   ```

2. Menghubungkan variabel file dengan nama file

   ```
   Assign (NmVar, nama file)
   ```

3. Menambah isi file

   ```
   Append(NmVar)
   ```

4. Menampilkannya di layar

   ```
   Write / writeln (NmVar, data item1, data item 2, …)
   ```

5. Menutup file

   ```
   Close (NmVar);
   ```

Contoh :

```
Program menambah_isi_file_HALLO_TXT;
Uses crt;
Var var_teks:TEXT;
     Pesan:string;
Begin
    Clrscr;
    Assign(var_teks,'Hallo.txt');
    Append(var_teks);
    Writeln(var_teks,'Ini penambahan data file hallo.txt!');
    Writeln(var_teks,'Contoh file teks');
    Writeln(var_teks,'----------------------------------------');
    Close(var_teks);
End.
```

1.5. Metode Pembacaan file oleh Turbo Pascal

- Menggunakan statemen operasi READ / READLN
- Bergantung pada tipe data variabel yang digunakannya :
  - Variabel berjenis numerik (byte,integer, real)
    - Tanda batas akhir pembacaan variabel jika ditemui blank(spasi), EOL (end of line), EOF (End of File) atau TAB.
    - Jika sebelum membaca data (atau dengan kata lain tidak ada datanya) pointer pascal menemukan EOL / EOF, maka variabel tersebut akan diisi NOL.
    - Jika string yang dibaca oleh variabel numerik tidak sah, maka terjadi kesalahan I/O : *Invalid numerik format.*
  - Variabel berjenis string

Karakter-karakter file aakan dibaca sampai ditemui EOL / EOF tercapai atau lebih besar dari ukuran stringnya.

- ♦ Variabel berjenis Char

  Yang dibaca hanya satu karakter saja.

- ❑ Agar pembacaan file sesuai dengan yang diharapkan, kadangkala perlu kita deklarasikan secara jelas jumlah byte yang disediakan untuk suatu variabel, atau mesti kita selipkan pencetakan spasi diantara dua variabel yang akan dibaca.
- ❑ Perlu diperhatikan kesesuaian tipe data, antara yang dituliskan dengan metode WRITE/WRITELN dan dengan yang akan kita baca.

2. File Bertipe

2.1. Karakteristik

- Berisi data format biner, ukurannya lebih kecil dari file teks.
- Tiap record memiliki tipe dan panjang yang sama. Bisa saja memiliki berbagai tipe asalkan dikelompokkan dalam RECORD.
- Dapat diakses secara random, elemen-elemennya bisa dibaca secara acak yang keberapa saja.

2.2. Membuat file Bertipe

Urutan Prosesnya :

1. Mendeklarasikan variabel file

   ```
   Var NmVar:FILE OF TypeVariabel;
   ```

   Dengan :

   NmVar : Nama variabel file bertipe

   TypeVariabel : Char, variabel tipe RECORD, variabel tipe array, real, variabel

array tipe record. Untuk satu file satu tipe elemen.

Contoh :

```
Type DaftarBarang      = Array [1..100] of integer;
     DataKonsumen      = RECORD
                           Nama      :string[15];
                           Alamat    :string[30];
                           Kode      :1..3;
     DaftarKonsumen    = Array [1..100] of Datakonsumen ;
Var
FileBarang      : File of Daftarbarang;
FileJumlah      : File of  integer;
FileData   : File of DataKonsumen;
FileKode   : File of Char;
```

2. Menghubungkan variabel file dengan nama file

   Assign (NmVar, nama file);

   Dengan :

   NmVar : Nama variabel file bertipe

   nama file : Nama file dalam bentuk string, format 8:3 penamaan dos, ditulis

   dalam bentuk string.

3. Membuat /membuka file bertipe

   Rewrite(NmVar); => untuk membuat

   Reset(NmVar); => untuk membuka

4. Menulis / membaca file Bertipe

Write (NmVar, data item1, data item 2, …) => untuk menulis

Read (NmVar, data item1, data item 2, …) => untuk membaca

Data item1, data item 2 dan seterusnya, harus berupa variabel, tidak bisa dituliskan secara langsung dalam bentuk konstanta. Variabelnya harus sama dengan deklarasi tipe file-nya.

Fungsi Writeln dan Readln tidak dapat digunakan pada file bertipe.

5. Menutup file

   Close (NmVar);

2.3. Fungsi-fungsi yang digunakan dalam file Bertipe

Seek (VarFile,N);

Menempatkan pointer ke posisi record ke-N

Contoh :

Seek(namafile,4); {pointer akan menunjuk posisi record ke-4, dengan nomor record 3}

FilePOS (VarFile);

Untuk menunjuk nomor record (nomor record dimulai dari record 0)

Contoh :

Posisi:=Filepos(Varfile); {mengetahui posisi pointer aktual di record mana, dan hasilnya diletakkan pada variabel posisi}.

Filesize(VarFile);

Mengukur besar file bertipe, yaitu mengetahui jumlah record yang berada dalam suatu file (jika file baru dibuat = 0)

Contoh :

```
..
Begin
  Write('input nomor record yang ingin dilihat');readln(NoRec);dec(NoRec)
  If NoRec >=filesize(VarFile) then
  Writeln('Nomor record terlalu besar');
  Else
     begin
     seek(VarFile,NoRec);  {Pointer menuju nomor record yang dimaksud}
     Read(VarFile,data);
     End;
..
end.
```

EOF(VarFile);

Untuk menunjuk akhir dari file.

Truncate(VarFile);

Untuk menghapus sebagian file

Contoh :

```
Seek(varfile,5);     {pointer menuju record nomor 5}
Truncate(VarFile);          {menghapus mulai record nomor 5 sampai habis }
```

Contoh Program File Bertipe :

```
Program menambah_dan_membuat_file_bertipe
Uses crt;
Type mhs = Record
              NPM : string[8];
              Nama: string[25];
              Alamat : string[20];
```

```
                End;
Var vfilemhs  : FILE of mhs;
    Recmhs    : mhs;
    I         : integer;
    Oke       : char;
Begin
  Clrscr;
  Assign (vfilemhs,'Dataku.dat');
  {$I-}reset(vfilemhs);{$I+}
  if IOResult <>0 then rewrite(vfilemhs);
  I:=filesize(vfilemhs);
  With recmhs do
  Begin
     Write('NPM :');readln(NPM);
     While NPM <>' ' do
     begin
        Write('Nama :');readln(Nama);
        Write('Alamat :');readln(Alamat);
        repeat
        Write('Save file…(Y/N)?');readln(oke);
        Until oke in ['Y','y','n','N'];
        If oke in ['Y','y'] then
        Begin
             Seek(vfilemhs,I);
             Write(vfilemhs,recmhs);
             I:=I+1;
        End;
        Writeln;
        Write('NPM:');readln(NPM);
     End;
  End;
```

```
  Close(vfilemhs);

End.

Program_Melihat_file_bertipe
Uses crt;
Type mhs = Record
NPM : string[8];
Nama: string[25];
Alamat : string[20];
End;

Var vfilemhs: FILE of mhs;
Recmhs:mhs;
I: integer;

Begin
    Clrscr;
    Assign(vfilemhs,'dataku.dat');
    Reset(vfilemhs);
    Writeln('------------------------------------------------------------------------');
    Writeln('No.        NPM          Nama              Alamat          ');
    Writeln('------------------------------------------------------------------------');

    With recmhs
    Begin
        For I:=1 to filesize(vfilemhs) do
        Begin
            Seek(vfilemhs),I-1);
            Read(vfilemhs),recmhs);
            Writeln(I:2,' ',NPM:8,Nama:25,Alamat:20);
```

```
        End;
    End;
    Close(vfilemhs);
End.
```

3. File Tidak Bertipe

3.1. Karakteristik

- File yang mengakses langsung ke media penyimpanan tanpa adanya pengenalan record dan sebagainya.
- Digunakan untuk tugas-tugas yang berhubungan dengan file biner yang dapat diproses tanpa mengenal jenis recordnya.

3.2. Membuat file Tidak Bertipe

Urutan Prosesnya :

1. Mendeklarasikan variabel file

   ```
   Var NmVar:FILE;
   ```

2. Menghubungkan variabel file dengan nama file

   ```
   Assign (NmVar, nama file);
   ```

   Dengan :

   NmVar : Nama variabel file bertipe

   nama file : Nama file dalam bentuk string, format 8:3 penamaan dos, ditulis dalam bentuk string.

3. Membuka file bertipe

Rewrite(NmVar[,*brec*]); => untuk membuat

Reset(NmVar[,*brec*]); => untuk membuka

Dengan :

Brec : Menunjukkan besar file dalam byte, opsional, boleh ditulis, boleh tidak,
dan besarnya kita tentukan sendiri. Defaultnya 128 Byte.

4. Menulis / membaca file tidak Bertipe

Blockwrite (NmVar, Buffer, jumlah[,jumtulis]) => untuk menulis

Dengan :

Buffer : daerah penyimpanan data yang akan ditulis ke dalam file. Buffer dapat
berupa suatu variabel dengan tipe apa saja sesuai dengan ukuran data yang akan ditulis di file.

Jumlah : jumlah data yang akan ditulis ke file dalam ukuran byte.

Jumtulis : suatu parameter yang tidak tetap yang boleh digunakan dan boleh juga
tidak digunakan, bila digunakan akan berisi jumlah byte yang dapat ditulis ke file.

Read (NmVar, data item1, data item 2, …) => untuk membaca

Dengan :

Buffer : daerah penyimpanan yang tipe variabelnya disesuaikan dengan jumlah
data yang dibaca.

Jumlah : jumlah byte yang akan dibaca dari file, dapat merupakan suatu variabel

dengan tipe word. Jumlah harus sama dengan besar buffer yang diberikan dan tidak boleh lebih dari 64 Kilobyte.

Jumbaca : merupakan variabel yang berisi laporan jumlah byte yang dapat dibaca dari file.

Contoh File tidak bertipe ::

```
Program menyalin_file
Uses crt;
Var
    Fileasal, filetuju  :file;
    Besar               :real;
    Buf                 :array [1..10240] of char;
    Hbaca, Hatulis      :word;
    nfile1,nfile2       :string;

Begin
  besar:=0;
  clrscr;
  write (‘Input nama file asal     :’);readln(nfile1);
  write (‘Input nama file tujuan :’);readln(nfile2);
  {$I-}
  Assign(fileasal,nfile1);
  Assign(filetuju,nfile2);
  Reset(fileasal,1);
  If IOResult <>0 then
```

```
      Begin
        Writeln('File asal tidak ada !');
        Halt(0);
      End;
  Rewrite(filetuju,1);
  If IOResult <>0 then
      Begin
        Writeln('File tujuan tidak bisa dibuat !');
        Halt(0);
      End;
  {$I+}
   Repeat
   Blockread(fileasal,buf,10240,hbaca);
   Besar:=besar + hbaca;
   Blockwrite(filetuju,buf,hbaca,htulis);
   If hbaca <> htulis then
      Begin
        Writeln('Tujuan tidak bisa menampung besar file');
        Halt(0);
      End;
   Until hbaca <>10240;
   Close (fileasal);
   Close (filetuju);
   Writeln ('Besar file yang dipindahkan :',besar:10:0);
End.
```

4. Menangani Kesalahan I/O pada File

Beberapa operasi pada file yang dapat mengalami kesalahan I/O :

Reset, rewrite, read, readln, write, writeln, close, append, seek

Agar program tidak berhenti, maka harus digunakan Compiler Directive Fungsi I/O result
Untuk pengecekan I/O file tersebut.

Bentuk Umum :
{$ Kode Kondisi}
Dengan :

- Kode : Karakter kode, untuk fungsi I/O result adalah I.
- Kondisi : OFF (-) / ON (+)
- Secara default {$I+} => ON
- I/O Result => bernilai 0 jika operasi berhasil
- Isi nilai selalu dihapus sendiri setiap kali dipanggil
- Penulisannya mengapit operasi-operasi yang rentan kesalahan

Contoh :

```
…
Assign(VarFile, ‘data.dat’);
{$I-}
Reset(VarFile);
{$I+}
fileada := IOResult = 0;
if fileada then
begin
writeln(‘file sudah ada…’);
…
end
else                              { Kondisi bila file tidak ada }
   begin
   rewrite(VarFile);              { Buat file baru }
```

```
    …
    write (VarFile, var1, var2, …);
    …
    close(VarFile);
    end;
…
```

5. Manipulasi File

5.1. Parameter

Parameter disebut juga command tail, merupakan informasi tambahan bagi program utama yang dapat diketikkan sesuai keperluan, untuk menjalankan rutin khusus dalam program tersebut.

Contoh : program scan.exe

Biasanya dituliskan pada prompt :

C:\scan a: /clean

String ‘a:’ dan string ‘/clean’ merupakan parameter, yang akan membuat scan program ke drive a dan kemudian menjalankan prosedur CLEAN.

Ada dua perintah untuk membuat parameter :

- **ParamCount** : untuk mengetahui jumlah parameter yang diberikan

  Bentuk Umumnya :

  ```
  Var_word := ParamCount;
  ```

- **ParamString** : untuk mengambil parameter tersebut dalam bentuk string

  Bentuk Umumnya :

  ```
  Var_string := ParamString(index);
  ```

5.2. Atribut File

Atribut file adalah satu byte data yang berisi keterangan dari suatu file. Karena bit –nya merupakan kode biner, maka dibuat lambang untuk mewakili bit-bit tersebut, sebagai berikut :

Lambang	Angka
ReadOnly	1
Hidden	2
SysFile	4
VolumeId	8
Directory	16
Archive	32

Ada 2 perintah yang berhubungan dengan atribut file, yaitu :

- **GetFAttr (Get File Atribut)** : Mengambil informasi atribut file.

  Bentuk umumnya :

  ```
  GetFAttr(nm_var,attr);
  ```

- **SetFAttr (Set File Atribut)** : Mengeset atribut file.

  Bentuk umumnya :

  ```
  GetFAttr(nm_var,attr);
  ```

Dengan :

Nm_var : nama suatu file yang telah di-assign dengan tipe apa saja.

Attr : Bilangan atribut yang akan diambil/diketahui dari suatu file.

Catatan :

1. File yang akan dilihat atau diset atributnya mesti di-assign terlebih dahulu.
2. Kedua perintah ini berada pada unit DOS dan WINDOWS Turbo Pascal, maka perlu menggunakan ‘Uses DOS’ atau ‘Uses Windows;’ pada deklarasi unit.

5.3. Menghapus File

Untuk menghapus suatu file, file tersebut harus di-assign terlebih dahulu. Bentuk umum perintahnya :

```
Erase(nm_var);
```

Contoh :

```
Program hapus;
Uses crt;
Var f:file;
Begin
  Clrscr;
  Assign(f,paramstr(1));
  Write( ‘File ‘,paramstr(1));
  {$I-}
  erase(f);
  {$I+}
  if IOResult <> 0 then writeln(‘Gagal mengapus’);
  else writeln(‘Menghapus sukses …’);
end.
```

5.4. Mengubah nama file

Untuk mengubah nama file, file tersebut harus di-assign terlebih dahulu. Bentuk umum perintahnya :

```
Rename(nm_var,nm_baru);
```

Contoh :

```
Program Ubah_nama;
Uses crt;
Var f:file;
Begin
  Clrscr;
  Assign(f,paramstr(1));
  Write( 'File ',paramstr(1));
  {$I-}
  rename(f,ParamStr(2));
  {$I+}
  if IOResult <> 0 then writeln('Gagal mengubah nama');
  else writeln('Mengubah nama menjadi ', ParamStr(2));
end.
```

5.5. Tanggal dan waktu File

Guna memanipulasi tanggal dan waktu file, Turbo Pascal menyediakan variabel khusus bernama DateTime dengan tipe data record dengan field-field sebagai berikut :

```
Type datetime = record;
      Year, month,day,hour,min,sec:word;
      End;
```

Setelah dideklarasikan, baru kita dapat mengambil dan menset informasi date dan time dengan menggunakan perintah :

- **GetFTime** : mengambil informasi tanggal dan jam file. Bentuk Umum :

GetFTime(nm_var,waktu);

- **SetFTime** : mengeset informasi tanggal dan jam file. Bentuk Umum :
  SetFTime(nm_var,waktu);

Dengan :

Nm_var : variabel file yang telah di-assign.

Waktu : waktu yang akan diambil (dalam bentuk LongInt).

Keterangan :

1. Pada saat GetFTime, variabel waktu masih merupakan tipe data sistem(LongInt), maka perlu diubah terlebih dahulu sebelum ditampilkan, yaitu dengan perintah :
   UnpackTime(waktu,var_datetime);
   Waktu : dideklarasikan dalam bentuk longint
   Var_datetime : variabel dari record datetime.

2. Sebelum melakukan SetFTime, variabel waktu masih merupakan variabel dari record datetime, maka perlu dikembalikan ke tipe data sistem(LongInt), yaitu dengan perintah :
   PackTime(var_datetime, waktu);
   Var_datetime : variabel dari record datetime.
   Waktu : dideklarasikan dalam bentuk tipe data sistem (longint).

3. Harus menggunakan deklarasi unit ‘Uses DOS;’ atau ‘Uses WinDOS;’

5.6. Mencari File

Menggunakan perintah :

```
Findfirst(path,attr,var_searchrec)
;
FindNext(var_searcRec);
```

Dengan :

path : file yang akan dicari, dapat diisi dengan wildcard seperti ‘*’ dan ‘?’. Pada unit

DOS menggunakan tipe string. Harus deklarasi unit ‘Uses Dos;’.

Attr : atribut file yang dicari

Var_searchrec : variabel dengan tipe searchrec sebagai berikut ini :

```
Type
Searchrec = record
            Fill : array [1..2] of char;
            Attr : byte;
            Time : longint;
            Size : longint;
            Name : string;
            End;
```

Contoh :

```
Program cari;
Uses crt,dos;
Var dirinfo :searchrec;
Begin
  Clrscr;
  Findfirst(paramstr(1),archive,dirinfo);
  Writeln (‘File-file yang dicari adalah :’);
  While doserror=0 do
  Begin
    Write(dirinfo.name:15, ‘ ‘);
```

```
    Writeln(dirinfo.size:10);
    Findnext(dirinfo);
  End;
End.
```

5.8. Mengecek Keberadaan File

Menggunakan perintah :

- **Fsearch** : untuk deklarasi unit DOS

  Bentuk :

  ```
  Var_pathstr := Fsearch(path,dirlist)
  ```

  Dengan :

  Path : file yang akan dicari

  Dirlist : letak dimana akan dicari yang didefinisikan di path tersebut.

- **FileSearch** : untuk deklarasi unit Windows

  Bentuk :

  ```
  FileSearch(var_string1, var_string2, var_string3)
  ```

  Dengan :

  Var_string1 : variabel tempat menaruh hasil proses.

  Var_string2 : variabel tempat menaruh file yang dicari.

  Var_string3 : variabel tempat melakukan pencarian file.

5.9. Memberikan directory File

Menggunakan 2 perintah di bawah ini:

- **FExpand**

  Bentuk umum :

  ```
  Var_pathstr1 := FExpand(Var_pathstr2);
  ```

Dengan :

Var_pathstr1 : hasil dari proses dengan penulisan file secara panjang

Var_pathstr2 : nama file yang akan diperpanjang penulisannya

Fungsi ini tidak mengecek keberadaan file yang dimanipulasinya. Bila file tidak ada, maka tetap akan menghasilkan path yang sedang aktif.

- **FileExpand**

Bentuk umum :

```
FileExpand(hasil,file_asal);
```

Dengan :

Hasil : merupakan hasil proses.

File_asal : nama file yang akan diperpanjang penulisannya.
