

Файлтай ажиллах талаар товчхон

Сэдвүүд:

- Файлтай ажиллах хэлбэрүүд
- Урсгалаар дамжуулан файлтай ажиллах
- Урсгал нээх ба хаах
- Файл руу бичих, файлаас унших

Файлтай ажиллах хэлбэрүүд

Бид оролт, гаралтын үйлдэл бүхий програм бичихдээ дэлгэц, гар гэсэн оролт, гаралтын хэрэгслүүдтэй ажиллаж байсан билээ. Програмын төвшинд дэлгэц, гар болон бусад оролт, гаралтын хэрэгслүүдийг байтуудын дараалал (byte sequence) мэтээр авч үздэг.

Нөгөө талаас, гадаад санах ой тухайлбал диск дээр орших файл бол байтуудын дараалал байдаг. Үүнтэй холбоотойгоор, оролт, гаралтын хэрэгсэл гэсэн ойлголтод мөн файлыг хамруулж болох ажээ.

Эсрэгээр, дээр дурдсан оролт, гаралтын хэрэгслүүдийг (дэлгэц, гар г.м.) бас файл гэж үзэж болох юм.

Ингэснээр, файлтай ажиллахдаа яг л оролт, гаралтын хэрэгсэлтэй ажиллаж буй мэтээр харьцаж болно. Ө.х. файл руу өгөгдөл бичих (гаралтын үйлдэл), файлаас өгөгдөл унших (оролтын үйлдэл) боломжтой.

Си хэлэнд өөрт нь файлтай ажиллах хэрэгслүүд байдаггүй. Оролт, гаралтын бүх үйлдлийг Си хэлний функцын сангийн (library) тусламжтай гүйцэтгэдэг. Функцын сан нь файлтай дараах хоёр хэлбэрээр ажиллах боломжийг олгоно:

- Доод төвшний хандалт (Low-level access)
- Урсгалаар дамжуулан ажиллах (Stream-level access)

Гэтэл функцын сан бол стандарт ба стандарт бус гэсэн хэсгүүдээс тогтдог. Стандарт бус хэсэг нь ANSI стандартад үл орох учраас хөрвүүлэгч (ө.х. програмчлалын систем) болгонд өөр өөр байх магадлалтай.

Доод төвшний хандалтыг хийхдээ стандарт бус функцын сангаас ашигладаг бол харин Урсгалаар дамжуулан файлтай ажиллахдаа стандарт функцын санг ашигладаг.

Тухайлбал Доод төвшний хандалт нь үйлдлийн системийн оролт, гаралтын хэрэгслийг шууд ашигладгаар онцлогтой. Харин эдгээр хэрэгсэл нь үйлдлийн систем бүрийн хувьд өөр өөр байж болно. Ө.х. хэрэгслүүдтэй ажиллагч функцууд нь өөр өөр байж болно. Үүнтэй холбоотойгоор, нэг үйлдлийн системд зориулан бичсэн програмыг өөр үйлдлийн систем рүү зөөхөд, ажиллахгүй байх талтай. Энэ шалтгаанаар Доод төвшний хандалтын талаар дурдалгүй орхиё.

Урсгалаар дамжуулан файлтай ажиллах

Урсгал (Stream) гэж юу вэ? Урсгал гэдэг нь ерөөсөө, ямар нэг эх үүсвэрээс ямар нэг хүлээн авагч руу мэдээлэл (өгөгдөл) дамжих процессыг илэрхийлсэн хийсвэр ойлголт юм. Арай тодорхой хэлбэл, мэдээллийг үүсгэж буй эсвэл мэдээллийг хүлээн авч буй бодит хэрэгслийг төлөөлсөн логик хэрэгсэл (logical device) юм.

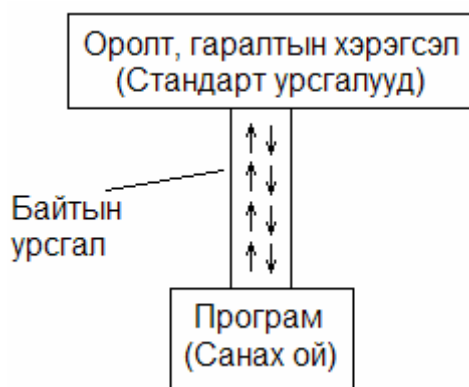
Ямар ч гадаад төхөөрөмжийг урсгалтай холбож (ө.х. урсгалаар илэрхийлж) болно. Ингэснээр, дэлгэц, гар г.м. оролт-гаралтын хэрэгсэл, эсвэл диск дээр байрлах физик файлтай шууд харьцаж ажиллахын оронд тэдгээр тус бүрийг төлөөлсөн урсгалтай ажиллана гэсэн үг.

Тиймээс, хэрэв програмд гараас оролтын үйлдэл хийж байвал энэ нь гарыг төлөөлсөн урсгалтай ажиллаж байгаа хэрэг болно. Хэрэв програм дэлгэцэнд гаралтын үйлдэл хийж байвал энэ нь дэлгэцийг төлөөлсөн урсгалтай ажиллаж байгаа хэрэг болно. Гар ба дэлгэцийг төлөөлсөн урсгалуудыг стандарт урсгалууд (standard streams) гэдэг. Стандарт урсгалуудыг дотор нь:

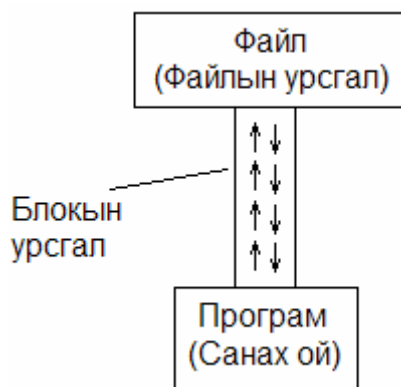
- Стандарт оролт (standard input) – гарыг төлөөлсөн урсгал
- Стандарт гаралт (standard output) – дэлгэцийг төлөөлсөн урсгал гэж хуваадаг.

Хэрэв програм файлтай ажиллаж байвал энэ нь файлыг төлөөлсөн урсгалтай ажиллаж буй хэрэг болно. Ийм урсгалыг файлын урсгал (file stream) гэдэг.

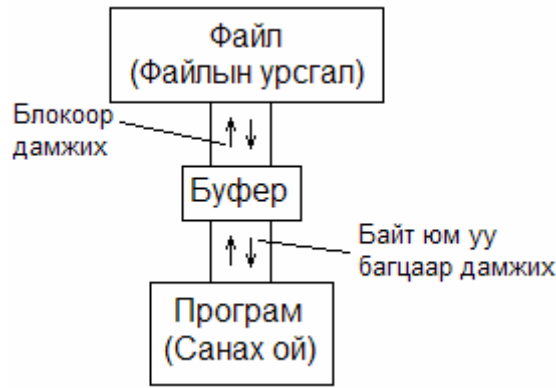
Програм ба стандарт урсгалын хооронд мэдээлэл байт байтаар дамждаг. Ө.х. солилцож буй мэдээллийн хамгийн бага хэмжээ нь байт байдаг байна.



Харин файлын урсгалтай ажиллаж буй тохиолдолд мэдээлэл багц багцаар дамждаг байна. Нэг ийм багц нь 512, эсвэл 1024 байтын хэмжээтэй байдаг. Үүнийг блок гэдэг. Учир нь аливаа файл диск дээр нэг буюу түүнээс олон блок зайг эзэлж оршдог.



Үнэн хэрэгтээ файлын урсгалтай ажиллах процесс буфер (buffer) хэмээн нэрлэгдэх завсрын санах ойг дамжин явагддаг байна. Програмаас файл руу (урсгал руу) бичиж буй тохиолдолд, мэдээлэл эхлээд буферт хуримтлагдсаар, тэр нь дүүрмэгц нэг бүтэн блокоор ганц удаагийн хандалтаар шууд диск рүү (урсгал руу) шилжинэ. Програм файлаас (урсгалаас) уншиж буй тохиолдолд, буфер бүрэн суларсан үед блок мэдээлэл буферт орж ирээд, тэндээс харин байт байтаар эсвэл тодорхой багцаар уншигдана. Буфер нь мэдээллийг түрхэн зуур хадгалах зориулалттай, үндсэн санах ойн тусгай хэсэг юм. Тиймээс буферээр дамжуулан мэдээлэл солилцох нь урсгалтай шууд харьцсанаас илүү хурдан явагддаг ажээ.



Ийнхүү урсгал хэмээх ойлголт оруулж ирэн, файлтай мэдээлэл солилцох процессыг хийсвэрлэснээр үйлдлийн системээс үл хамаарах давуу тал бий болдог. Ингэснээр тухайн програм нэг системээс нөгөө рүү (эсвэл нэг төрлийн компьютерээс нөгөө рүү) ямар нэг өөрчлөлтгүйгээр зөөгдөх боломжтой болж байгаа. Тийм ч учраас урсгалтай ажиллах хэрэгслүүд Си хэлний функцын сангийн стандарт хэсэгт агуулагддаг. Ө.х. ANSI стандартаар тодорхойлогдсон байна.

Стандарт сангийн функцуудыг ашиглахын тулд харгалзах толгой файлыг програмдаа оруулах ёстой болно. Энэ нь бидний мэдэх **stdio.h** файл юм:

```
#include <stdio.h>
```

Функцуудын тусламжтайгаар урсгал дээр дараах үйлдлүүдийг хийж болно:

- Урсгал нээх ба хаах (урсгалыг тодорхой файлтай холбох ба салгах)
 - Тодорхой өгөгдөл, тухайлбал тэмдэгт, тэмдэгт мөр, тоон утга, дурын урттай багц мэдээлэл г.м.-ийг оруулах/гаргах (унших/бичих)
 - Урсгалтай ажиллах үед гарах алдааг шинжлэх, урсгалын (файлын) төгсгөлд хүрсэн эсэхийг шалгах
 - Буферыг удирдах, буферын хэмжээг тохируулах
- г.м.

Урсгал нээх ба хаах

Програм нь файлтай ажиллахын тулд дараах үйлдлүүдийг хийх ёстой юм:

1. Урсгал үүсгэх
2. Урсгалыг нээж, файлтай холбох
3. Урсгал (файлтай) дээр оролт, гаралтын үйлдлийг хийх
4. Урсгалыг хаах (файлыг хаах)

Си хэлэнд урсгалыг урсгалын заагч хэмээх ойлголтоор илэрхийлдэг. Урсгалын заагч нь **FILE** гэсэн өгөгдлийн төрөлд хамаардаг. Тиймээс урсгал шинээр үүсгэхийн тулд урсгалын заагчийг зарлах хэрэгтэй юм. Зарлахдаа дараах загварын дагуу бичнэ:

```
FILE *урсгалын_нэр;
```

Энд:

- **FILE** – өгөгдлийн төрлийг заасан түлхүүр үг
- *урсгалын_нэр* – урсгалыг (урсгалын заагчийг) нэрлэсэн чөлөөт идентификатор.

Ж.нь:

```
FILE *mystream;
```

Энд *mystream* хэмээх урсгалыг үүсгэж байна.

Дараагийн алхам бол үүсгэсэн урсгалаа нээж, физик файлтай холбох явдал. Үүний тулд:

```
урсгалын_нэр = fopen ( файлын_бүтэн_нэр, урсгалыг_нээх_горим );
```

хэлбэрийн операторыг бичдэг. Энд:

- **fopen()** – урсгалыг нээж, файлтай холбох стандарт функц
- *файлын_нэр* – холбох файлын нэр

- *урсгалыг_нээх_горим* – урсгалтай (файлтай) ажиллах хэлбэр.

Ж.нь:

FILE *mystream;

mystream = **fopen** (*файлын_бүтэн_нэр, урсгалыг_нээх_горим*);

Файлын бүтэн нэр нь тэмдэгт мөр төрлийн өгөгдөл байх тул давхар хашилт дотор бичнэ. Файлын бүтэн нэр нь файлын нэр, өргөтгөл гэсэн хэсгүүдээс тогтох ёстой. Ж.нь:

mystream = **fopen** (“myfile.dat”, *урсгалыг_нээх_горим*);

Заримдаа файлын бүтэн нэрийг өргөтгөлгүй бичсэн байж болно:

mystream = **fopen** (“myfile”, *урсгалыг_нээх_горим*);

Урсгалыг нээх горим буюу урсгалтай ажиллах хэлбэр нь 6 төрөл байдаг. Урсгалыг нээж, файлтай холбохдоо эдгээрийн аль нэгийг сонгох ёстой. Эдгээр горим нь:

- “w” – нэр бүхий файлыг, зөвхөн бичихийн тулд шинээр үүсгэх горим
- “r” – нэр бүхий файлыг, зөвхөн уншихын тулд нээх горим
- “a” – нэр бүхий файлыг, төгсгөлд нь нэмэж бичихийн тулд нээх/шинээр үүсгэх горим
- “w+” – нэр бүхий файлыг, унших/бичихийн тулд шинээр үүсгэх горим. Файлын төгсгөлд нэмэж бичих боломжтой
- “r+” – нэр бүхий файлыг, унших/бичихийн тулд нээх горим. Файлын төгсгөлд нэмэж бичих боломжгүй
- “a+” – нэр бүхий файлыг, унших/бичихийн тулд нээх/шинээр үүсгэх горим. Файлын төгсгөлд нэмэж бичих боломжтой.

Эндээс харахад “+” тэмдэг бүхий горимууд нь урсгал руу бичих, урсгалаас унших боломжийг давхарт нь олгодог байна. Ж.нь:

mystream = **fopen** (“myfile.dat”, “w+”);

Энд myfile.dat файлыг (түүнийг төлөөлсөн урсгалыг) унших ч, бичих ч боломжтойгоор нээж байна. Хэрэв энэ файл диск дээр үүсээгүй байсан бол оператор биелсний дараа шинээр үүсэх болно.

Урсгал нээж, файлтай холбох үед дараах алдаанууд үүсч болно. Тухайлбал зөвхөн унших горимын үед бол “нэр бүхий файл олдсонгүй”, бичих горимын үед бол “диск дүүрсэн байна”, “диск бичүүлэхээс хамгаалагдсан байна” г.м. Ийм тохиолдлуудад **fopen()** функц NULL гэсэн утга буцаадаг. Ө.х. урсгал маань NULL гэсэн утгатай болдог байна. Харин бусад тохиолдолд (алдаагүй үед) бол урсгал ямагт NULL-ээс ялгаатай утгатай байдаг ажээ.

Урсгалыг нээж, файлтай холбоод шаардлагатай оролт, гаралтын үйлдлүүдийг хийсний дараа дахиж файлтай ажиллахгүй гэсэн тохиолдолд урсгалыг заавал хааж байх нь зүйтэй. Үүний тулд **fclose()** гэсэн стандарт функцийг ашиглана. Бичих загвар нь:

fclose (*урсгалын_нэр*);

гэсэн оператор байна. Ж.нь:

FILE *mystream;

mystream = **fopen** (“myfile.dat”, “w+”);

...

fclose (mystream);

/* урсгал үүсгэх */

/* урсгал нээж, файлтай холбох */

/* оролт, гаралтын үйлдлүүд */

/* урсгал хаах */

Жишээ програм

Дээр дурдсан бүхнийг нэгтгээд нэг жишээ програм үзье. Энэ нь зүгээр л хоёр өөр урсгал дараалан нээгээд, хааж буй жишээ юм:

#include <stdio.h>

main ()

{

FILE *stream1, *stream2;

stream1 = **fopen** (“myfile1.dat”, “w”);

stream2 = **fopen** (“myfile2.dat”, “w”);

/* энд оролт, гаралтын үйлдлүүд байна гэж бодъё */

fclose (stream1);

```
fclose (stream2);
}
```

Энэ програм ажилласны дараа диск дээр, програмын эх файл байрлах тэр газарт myfile1.dat, myfile2.dat гэсэн файлууд үүссэн байх болно. Урсгалууд руу ямар нэг зүйл бичээгүй тул файлууд хоосон байх болно.

Файл руу бичих, файлаас унших

Урсгал үүсгэж, нээж, файлтай холбосны дараагаар унших, бичих боломжтой болдог. Ингэж файлтай ажиллахад зориулагдсан дараах функцууд Си хэлний стандарт санд багтжээ. Үүнд:

- **fputc()** – файл руу ганц тэмдэгт бичих
- **fgetc()** – файлаас ганц тэмдэгт унших
- **fputs()** – файл руу тэмдэгт мөр бичих
- **fgets()** – файлаас тэмдэгт мөр унших
- **fprintf()** – файл руу өгөгдлийг (өгөгдлүүдийг) тодорхой загвараар бичих
- **fscanf()** – файлаас өгөгдлийг (өгөгдлүүдийг) тодорхой загвараар унших

г.м.

Эдгээрээс **fprintf()**, **fscanf()** функцуудыг хэрхэн ашиглахтай товч танилцъя. Энэ хоёр функц нь файлтай ажилладаг гэдгээрээ л бидний мэдэх **printf()**, **scanf()** функцуудээс ялгаатай.

Файл руу бичих

fprintf() функцыг ашигласан оператор:

fprintf (урсгалын_нэр, бичих_формат, бичих_өгөгдлүүд);

гэсэн загварын дагуу бичигдэнэ. Энд:

- *урсгалын_нэр* – файлыг төлөөлсөн урсгал
- *бичих_формат* – ямар төрлийн өгөгдлүүдийг яаж хэвлэхийг заасан тэмдэгт мөр
- *бичих_өгөгдлүүд* – файл руу бичигдэх өгөгдлүүд.

Ж.нь:

```
fprintf (mystream, "u", 5);
```

урсгал формат тогтмол

```
fprintf (stream1, "f", value);
```

урсгал формат хувьсагч

```
fprintf (stream2, "d f f", i, x, y);
```

урсгал формат хувьсагчууд

г.м. Энд, эхний оператор нь mystream гэсэн урсгал руу 5 гэсэн тоог **unsigned int** гэсэн бүхэл төрлийн өгөгдөл болгон бичиж байна. Хоёр дахь оператор нь stream1 гэсэн урсгал руу value гэсэн хувьсагчийн утгыг бодит **float** төрөл болгон бичиж байна. Харин гурав дахь оператор stream2 гэсэн урсгал руу бүхэл i, бодит x, y гэсэн өгөгдлүүдийг нэг мөрөнд, хоорондоо сул зайтай хэвлэгдэхээр бичиж байна. Энд, урсгал болгон ямар нэг файлыг төлөөлж буй гэдгийг битгий мартаарай.

Урсгал (файл) руу бичиж байх тохиолдолд, тэнд яг л дэлгэцийнх шиг, гэхдээ үл үзэгдэх ямар нэг курсор байна гэж ойлгох хэрэгтэй. Ийм курсорыг байршил заагч гэнэ. Байршил заагч нь урсгал дотор хаана байрлаж байна, тэнд өгөгдөл бичигдэнэ. Урсгалыг “r” юм уу “w” горимоор нээх үед байршил заагч нь урсгалын эхэнд очдог бол, “a” горимоор нээх үед урсгалын төгсгөлд очсон байдаг. Тиймээс форматыг бичихдээ удирдах кодууд, тухайлбал курсорыг шинэ мөрийн эхэнд шилжүүлэх тэмдэгтийг (“\n”) ашиглаж болно. Ж.нь:

```
fprintf (mystream, "%d\n", 1);
fprintf (mystream, "%d", 2);
```

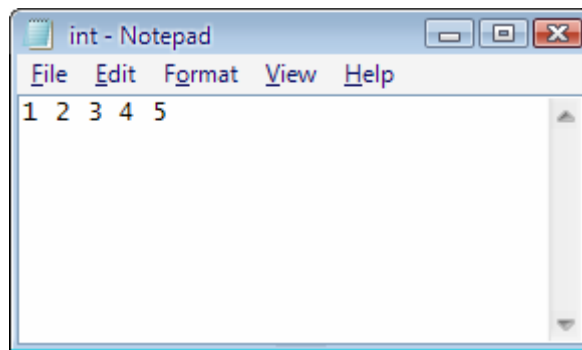
Энд, эхний оператор `mystream` хэмээх урсгал руу 1 гэсэн утгыг `int` форматаар бичээд, байршил заагчийг шинэ мөрийн эхэнд шилжүүлж байгаа юм. Хоёр дахь оператор нь 2 гэсэн тоог байршил заагчийн байгаа газар бичнэ. Иймээс файл дотор 1 ба 2 гэсэн тоонууд доош цуварч бичигдсэн байх болно.

Жишээ програм

Энэ програм нь `int.dat` гэсэн файл үүсгээд, түүн рүү 1-ээс 5 хүртэлх тоог нэг мөрөнд дундаа зайтайгаар цувуулан хэвлэнэ:

```
#include <stdio.h>
main ()
{
    FILE *fp;
    unsigned int i;
    fp = fopen ("int.dat", "w");
    for (i=1; i<=5; i++) fprintf (fp, "%u ", i);
    fclose (fp);
}
```

Програм ажилласны үр дүн болох `int.dat` файлыг харуулав:



Файлаас унших

`fscanf()` функцыг ашигласан оператор:

`fscanf` (*урсгалын нэр, унших формат, өгөгдлүүдийн хаягууд*);

гэсэн загварын дагуу бичигдэнэ. Энд:

- *урсгалын нэр* – файлыг төлөөлсөн урсгал
- *формат* – ямар төрлийн өгөгдлүүдийг яаж уншихыг заасан тэмдэгт мөр
- *өгөгдлүүдийн хаягууд* – файлаас уншиж буй утгуудыг авах өгөгдлүүдийн хаяг.

Уг функцыг ашиглахдаа дараах зүйлийг анхаарах хэрэгтэй. Өгөгдөл унших формат нь, файлд өгөгдлүүдийг анх бичсэн тэрхүү бичих форматтай адилхан байх хэрэгтэй.

Жишээ програм

Өмнөх жишээ програмын үүсгэсэн `int.dat` файлыг (доторхыг нь) дэлгэцэнд хэвлэн гаргая. Үүний тулд тэр файлаас өгөгдөл унших хэрэгтэй юм. Зөв уншихын тулд унших формат нь өгөгдлийг бичсэн форматтай адилхан байх хэрэгтэй. Харин түүнийг нь бид мэдэж байгаа билээ:

```
#include <stdio.h>
#include <conio.h>
main ()
{
    FILE *fp;
    unsigned int i, k;
    fp = fopen ("int.dat", "r");
    for (i=1; i<=5; i++)
    {
        fscanf (fp, "%u ", &k);
        printf ("%u ", k);
    }
}
```

```
}  
fclose (fp);  
getch ();  
}
```

Програм ажилласны үр дүнг харуулав:

