

ՀՏԴ 681/3

Ինֆորմատիկա

Սվետլանա ԵՐԻՅԱՆ

ԱրՊՀ կիրառական մաթեմատիկայի և ինֆորմատիկայի ամբիոնի դոցենտի պաշտոնակատար

e-mail. s.a_ericyan@mail.ru

Անուշ ՀԱՐՈՒԹՅՈՒՆՅԱՆ

ԱրՊՀ կիրառական մաթեմատիկայի և ինֆորմատիկայի ամբիոնի դասախոս

e-mail. asrian.anush@mail.ru

Վազգեն ԱՌՄՍՄՅԱՆ

ԱրՊՀ կիրառական մաթեմատիկայի և ինֆորմատիկայի ամբիոնի դոցենտ, մ.գ.թ.

e-mail. varustamyan@rambler.ru

ՖԱՅԼԵՐԻ ՄԻԱԶՈՒԼՄԱՆ ԽՆԴՐԻ ԻՐԱԿԱՆԱՑՄԱՆ ԲԱԶՄԱՀՈՍՔԱՅԻՆ ԾՐԱԳՐԻ ՍՏԵՂԾՈՒՄԸ ՆԵՐԱԾՄԱՆ/ԱՐՏԱԾՄԱՆ ԳՈՐԾՈՂՈՒԹՅՈՒՆՆԵՐԻ ԱՍԻՆԽՐՈՆ ԵՂԱՆԱԿԻ ՀԻՄԱՆ ՎՐԱ

Սույն հոդվածում ներկայացված է հավելված՝ Windows օպերացիոն համակարգերի կիրառական ծրագրավորման ինտերֆեյսի (API) միջոցով, որը միաձուլում է կարգավորված ֆայլեր:

Բանալի բառեր՝ ասինխրոն, բազմահոսքային, overlapped, ռեկուրսիա, ֆայլերի միաձուլում

СОЗДАНИЕ МНОГОПОТОЧНОЙ ПРОГРАММЫ СЛИЯНИЯ ФАЙЛОВ НА ОСНОВЕ АСИНХРОННЫХ ОПЕРАЦИЙ ВВОДА- ВЫВОДА

В статье представлено приложение, использующее интерфейс прикладного программирования Windows (API), цель которого слияние упорядоченных файлов.

Ключевые слова: асинхронный ввод-вывод, многопоточный, overlapped, рекурсия, слияние файлов.

CREATING A MULTI-STREAM FILE MERGER SOLUTION BASED ON ASYNCHRONOUS INPUT-OUTPUT OPERATIONS

This article provides an application that uses the Windows Application Programming Interface (API) that merges adjusted files.

Key words: asynchronous Input/Output, multithreading, overlapped, recursion, file merging.

Մշակված է հավելված Windows օպերացիոն համակարգերի կիրառական ծրագրավորման ինտերֆեյսի (API) միջոցով, որտեղ օգտագործված է ծրագրային հոսքեր, ասինխրոն ներածում/արտածում, ռեկուրսիա:

Օգտագործված է ծրագրի արդյունավետությունը և արագությունը ապահովելու համար այլ միջոցներ այնպիսին, ինչպիսին է օրինակ overlapped:

Ասինխրոն ծրագրավորման հիմնական գաղափարը կայանում է նրանում, որպեսզի իրականացվի մեթոդների առանձին կանչեր և առանց սպասելու այդ կանչերի ավարտին զուգահեռ իրականացնել մեկ այլ աշխատանք:

Մինխրոն գործարկման ժամանակ հավելվածն ունի միայն մեկ հոսք: Ասինխրոն ծրագրավորման օգնությամբ հնարավորություն ենք ստանում գործարկել բազմաթիվ զուգահեռ հոսքեր և դրանց գործարկման ընթացքում օգտագործողի կողմից սահմանել նոր գործողություններ[2]:

Ալգորիթմը նախատեսված է 2ⁿ քանակով ֆայլերի համար, այլապես բերել այդ դեպքի:

Խնդիրը կայանում էր հետևյալում՝ ստեղծել ծրագիր, որի միջոցով անհրաժեշտ է միաձուլել մի քանի տեսակավորված ֆայլերի պարունակությունները մեկ այլ ֆայլի մեջ տեսակավորված վիճակում՝ օգտագործելով ներածման/արտածման ասինխրոն մոտեցումներից որևէ մեկը: Տվյալ ծրագրի իրականացման համար ավելի էֆեկտիվ է օգտագործել բազմահոսքային սիմետրիկ մոդելի մեթոդը:

Ծրագիրը ստեղծված է հետևյալ ալգորիթմով: Ռեկուրսիայի առաջին մակարդակում ունենալով 2ⁿ քանակով ֆայլեր կատարում է զույգ առ զույգ միաձուլում իրականացնող բոլոր հոսքերը: Հաջորդ մակարդակում արդեն ունենք 2 անգամ քիչ քանակով ֆայլեր: Նույն պրոցեսը կրկնվում է այնքան մինչև ստացվի 1 ֆայլ, որը վերջնական պատասխանն է:

Այն մեծ կիրառություն ունի վիճակագրությունում՝ վերջնական որոշում կայացնելու համար, բարդ համակարգերի տեսության մեջ, մասնավորապես զանգվածային սպասարկան ոլորտում:

Ստորև ներկայացված է այդ ծրագրի կոդը:

```
// asynchrMThread.cpp
// asynchrMThread output_file input_file1 input_file2 ...

#include <tchar.h>
#include <windows.h>
#include <stdio.h>
#include <process.h>
#define _STATICLIB
#define _MT
#define BUF_SIZE 512
typedef struct _THREADARG{
HANDLE hIn1;
HANDLE hIn2;
HANDLE hOut;
} THREADARG;
static unsigned int WINAPI ThMergeFiles(void* pThArg);
int _tmain(int argc, _TCHAR* argv[])
{ struct TempFile{
    TCHAR TempFileName [MAX_PATH];
}*tempName;
HANDLE * hIn;
THREADARG *Tharg;
int i,iFile,nThs,nTemp;
for( i=1;((argc-2)>>i)>=2;i++);
nThs=i;
nTemp=argc-2-1;
Tharg=(THREADARG*) malloc( nTemp* sizeof(THREADARG));
tempName=(TempFile*)malloc(nTemp*sizeof(TempFile));
hIn=(HANDLE*)malloc((argc-2+ nTemp)*sizeof(HANDLE));
for (iFile =0;iFile<argc-2; iFile++)
hIn[ iFile]=CreateFile(argv[iFile +2],GENERIC_READ,0,NULL,OPEN_EXISTING,
0 ,NULL);
for (i=0;iFile< argc-2+nTemp-1; iFile++,i++)
{GetTempFileName(_T("."),_T("mrg"),0, tempName[i].TempFileName );
hIn [iFile]= CreateFile(tempName[i].TempFileName,
```

```

    GENERIC_WRITE|GENERIC_READ, 0,NULL,
    CREATE_ALWAYS,FILE_FLAG_DELETE_ON_CLOSE,NULL);
}
hIn[iFile]=CreateFile(argv[1],GENERIC_WRITE|GENERIC_READ,
,NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL);
int n=0,iTh,m,k=0;
unsigned int ThId;
int nThPass=(argc-2)/2;
m= argc-2;
HANDLE *ThreadHandle;
ThreadHandle=(HANDLE*)malloc(( nTemp)*sizeof(HANDLE));
while(nThPass>=1)
{ for( iTh=0;iTh <nThPass; iTh++)
{
    Tharg[iTh+k].hIn1= hIn [n +iTh*2];
    Tharg[iTh+k].hIn2= hIn [n +iTh*2+1];
    Tharg[iTh+k].hOut= hIn[m +iTh];
    ThreadHandle[iTh+k]=(HANDLE)_beginthreadex(NULL,0,
ThMergeFiles,(void*)&Tharg[iTh+k],0,&ThId);
}

WaitForMultipleObjects(nThPass, ThreadHandle+k,TRUE,INFINITE );
n+=m ;k+=nThPass; m+= nThPass; nThPass/=2;
}
int inBuffer1 [BUF_SIZE*5];DWORD nIn1;OVERLAPPED ov={0,0,0,0,NULL};
printf("test1.dat\n");
ReadFile (hIn[0],inBuffer1, BUF_SIZE*sizeof(int)*5,&nIn1,&ov);
for(i=0;i<nIn1/sizeof(int);i++)
{printf("%d ",inBuffer1[i]);
if((i+1)%15==0)
printf("\n");
}
printf("\n");
printf("test2.dat\n");
ReadFile (hIn[1], inBuffer1, BUF_SIZE*sizeof(int)*5,&nIn1,&ov);
for(i=0;i<nIn1/sizeof(int);i++)
{printf("%d ",inBuffer1[i]);

```

```
if((i+1)%15==0)
printf("\n");
printf("\n");
printf("test3.dat\n");
ReadFile (hIn[2], inBuffer1, BUF_SIZE*sizeof(int)*5,&nIn1,&ov);
for(i=0;i<nIn1/sizeof(int);i++)
{printf("%d ",inBuffer1[i]);
if((i+1)%15==0)
printf("\n");
}
printf("\n");
printf("test4.dat\n");
ReadFile (hIn[3], inBuffer1, BUF_SIZE*sizeof(int)*5,&nIn1,&ov);
for(i=0;i<nIn1/sizeof(int);i++)
{printf("%d ",inBuffer1[i]);
if((i+1)%15==0)
printf("\n");
}
printf("\n");
printf("mergeSortFiles.out\n");
ReadFile (hIn[argc-2+nTemp-1], inBuffer1, BUF_SIZE*sizeof(int)*5,&nIn1,&ov);
for(i=0;i<nIn1/sizeof(int);i++)
{printf("%d ",inBuffer1[i]);
if((i+1)%15==0)
printf("\n");}
printf("\n");
for(iTh=0;iTh < nTemp; iTh++)
    CloseHandle(ThreadHandle[iTh]);
for(iTh=0;iTh <argc-2+nTemp; iTh++)
    CloseHandle(hIn[iTh]);
free(tempName);
free(Tharg);
free(hIn);
free(ThreadHandle);
return 0;
}
```

```

static unsigned int WINAPI ThMergeFiles(void *pvThArg)
{
    OVERLAPPED ov={0,0,0,0,NULL};
    THREADARG*pThArg=(THREADARG*)pvThArg;
    int inBuffer1 [BUF_SIZE], inBuffer2[BUF_SIZE],outBuffer [ BUF_SIZE];
    LARGE_INTEGER FileSize1, FileSize2,FilePos;
    LONGLONG curPosIn1, curPosIn2, curPosOut;
    FilePos.QuadPart=0;
    FileSize1.LowPart=GetFileSize((THREADARG*)pThArg-
>hIn1,(LPDWORD)&FileSize1.HighPart);
    FileSize2.LowPart=GetFileSize( pThArg-
>hIn2,(LPDWORD)&FileSize2.HighPart);
    DWORD nIn1,nIn2,nOut;
    int i1,i2,i3;
    i1=i2=i3=0;
    curPosIn1=curPosIn2=curPosOut =0;
    BOOL ok1,ok2;
    ok1=ok2=TRUE;
    ReadFile (pThArg->hIn1, inBuffer1, BUF_SIZE*sizeof(int),&nIn1,&ov);
    curPosIn1+= nIn1;
    ReadFile (pThArg->hIn2, inBuffer2, BUF_SIZE*sizeof(int),&nIn2,&ov);
    curPosIn2+= nIn2;
    while(1){
        if ( inBuffer1[i1] < inBuffer2[i2])
        {
            outBuffer[i3]= inBuffer1[i1]; ++i3; ++i1;
            if ( i1* sizeof(int)== nIn1)
            {
                if(ReadFile( pThArg->hIn1, inBuffer1,
BUF_SIZE*sizeof(int),&nIn1,NULL)&&nIn1>0)
                {i1=0;curPosIn1+= nIn1;}
                else
                {ok1=FALSE;break;}
            }
        }
        else
        {

```

```
outBuffer[i3]= inBuffer2[i2]; i3++; i2++;
if(i2* sizeof(int)== nIn2)
{
    if(ReadFile( pThArg->hIn2, inBuffer2,
BUF_SIZE*sizeof(int),&nIn2,NULL)&&nIn2>0)
    {i2=0;curPosIn2+= nIn2;}
    else
    {ok2=FALSE;break;}
}
}
if (i3== BUF_SIZE)
{
    WriteFile(pThArg->hOut,outBuffer, BUF_SIZE*sizeof(int) ,&nOut,&ov);
    i3=0;
    FilePos.QuadPart+=nOut;
    ov.Offset=FilePos.LowPart;
    ov.OffsetHigh=FilePos.HighPart;
}
}
if (ok1==FALSE && ok2==TRUE)
{
    WriteFile(pThArg->hOut,outBuffer, i3*sizeof(int) ,&nOut,NULL);
    curPosOut+=nOut;
    WriteFile(pThArg->hOut,inBuffer2+i2, nIn2-i2*sizeof(int) ,&nOut,NULL);
    curPosOut+=nOut;
    while(ReadFile( pThArg->hIn2, inBuffer2,
BUF_SIZE*sizeof(int),&nIn2,NULL)&&nIn2>0)
    {
        curPosIn2+= nIn2;
        WriteFile (pThArg->hOut,inBuffer2,nIn2,&nOut,NULL);
        curPosOut+=nOut;
    }
}
if (ok1== TRUE && ok2== FALSE)
{
    WriteFile(pThArg->hOut,outBuffer, i3*sizeof(int) ,&nOut,NULL);
    curPosOut+=nOut;
```

```

WriteFile(pThArg->hOut,inBuffer1+i1, nIn1-i1*sizeof(int) ,&nOut,NULL);
curPosOut+=nOut;
while(ReadFile( pThArg->hIn1, inBuffer1,
BUF_SIZE*sizeof(int),&nIn1,NULL)&&nIn1>0)
{
    curPosIn1+= nIn1;
    WriteFile (pThArg->hOut,inBuffer1, nIn1 ,&nOut,NULL);
    curPosOut+=nOut;
}
}
_endthreadex(0);
return 0;
}

```

Գործարկելով այս ծրագիրը՝ որպես արդյունք կունենանք հետևյալը՝

```

Администратор: C:\Windows\system32\cmd.exe
C:\asynchMT\thread_mergeSortFiles.out test1.dat test2.dat test3.dat test4.dat
test1.dat
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56
60 64 68 72 76 80 84 88 92 96 100 104 108 112 116
120 124 128 132 136 140 144 148 152 156 160 164 168 172 176
180 184 188 192 196 200 204 208 212 216 220 224 228 232 236
240 244 248 252 256 260 264 268 272 276 280 284 288 292 296
300 304 308 312 316 320 324 328 332 336 340 344 348 352 356
360 364 368 372 376 380 384 388 392 396
test2.dat
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57
61 65 69 73 77 81 85 89 93 97 101 105 109 113 117
121 125 129 133 137 141 145 149 153 157 161 165 169 173 177
181 185 189 193 197 201 205 209 213 217 221 225 229 233 237
241 245 249 253 257 261 265 269 273 277 281 285 289 293 297
301 305 309 313 317 321 325 329 333 337 341 345 349 353 357
361 365 369 373 377 381 385 389 393 397
test3.dat
2 6 10 14 18 22 26 30 34 38 42 46 50 54 58
62 66 70 74 78 82 86 90 94 98 102 106 110 114 118
122 126 130 134 138 142 146 150 154 158 162 166 170 174 178
182 186 190 194 198 202 206 210 214 218 222 226 230 234 238
242 246 250 254 258 262 266 270 274 278 282 286 290 294 298
302 306 310 314 318 322 326 330 334 338 342 346 350 354 358
362 366 370 374 378 382 386 390 394 398
test4.dat
3 7 11 15 19 23 27 31 35 39 43 47 51 55 59
63 67 71 75 79 83 87 91 95 99 103 107 111 115 119
123 127 131 135 139 143 147 151 155 159 163 167 171 175 179
183 187 191 195 199 203 207 211 215 219 223 227 231 235 239
243 247 251 255 259 263 267 271 275 279 283 287 291 295 299
303 307 311 315 319 323 327 331 335 339 343 347 351 355 359
363 367 371 375 379 383 387 391 395 399
mergeSortFiles.out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99 100 101 102 103 104
105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
120 121 122 123 124 125 126 127 128 129 130 131 132 133 134
135 136 137 138 139 140 141 142 143 144 145 146 147 148 149
150 151 152 153 154 155 156 157 158 159 160 161 162 163 164
165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 187 188 189 190 191 192 193 194
195 196 197 198 199 200 201 202 203 204 205 206 207 208 209

```

```

Администратор: C:\Windows\system32\cmd.exe
303 307 311 315 319 323 327 331 335 339 343 347 351 355 359
363 367 371 375 379 383 387 391 395 399
mergeSortFiles.out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99 100 101 102 103 104
105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
120 121 122 123 124 125 126 127 128 129 130 131 132 133 134
135 136 137 138 139 140 141 142 143 144 145 146 147 148 149
150 151 152 153 154 155 156 157 158 159 160 161 162 163 164
165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 187 188 189 190 191 192 193 194
195 196 197 198 199 200 201 202 203 204 205 206 207 208 209
210 211 212 213 214 215 216 217 218 219 220 221 222 223 224
225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
240 241 242 243 244 245 246 247 248 249 250 251 252 253 254
255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
270 271 272 273 274 275 276 277 278 279 280 281 282 283 284
285 286 287 288 289 290 291 292 293 294 295 296 297 298 299
300 301 302 303 304 305 306 307 308 309 310 311 312 313 314
315 316 317 318 319 320 321 322 323 324 325 326 327 328 329
330 331 332 333 334 335 336 337 338 339 340 341 342 343 344
345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
375 376 377 378 379 380 381 382 383 384 385 386 387 388 389
390 391 392 393 394 395 396 397 398 399
C>

```

Windows-ում ներածման/արտածման գործողությունների ասինխրոն իրականացումն ապահովվում է հետևյալ երեք մեթոդներով [1]՝

- բազմահոսքային ներածում/ արտածում (**Multithreaded I/O**): Պրոցեսի կամ պրոցեսների խմբի մեջ յուրաքանչյուր հոսք իրականացնում է սովորական սինխրոն ներածում/արտածում, իսկ այդ ժամանակ մնացած հոսքերը կարող են շարունակել իրենց գործարկումները:

- փոխաձածկվող ներածում/արտածում (**Overlapped I/O**): Իրականացնելով ընթերցման, գրառման կամ ինչ-որ այլ ներածման/արտածման գործողություն՝ հոսքը շարունակում է իր գործարկումը: Եթե գործարկման շարունակման համար հոսքին անհրաժեշտ է ներածման/արտածման արդյունքները, ապա այն սպասում է մինչև համապատասխան դեսկրիպտորը դառնա հասանելի կամ մինչև չհասնի նշված պատահարին (**event**):

- ավարտման պրոցեդուրաներ (կամ ընդլայնված ներածում/արտածում) (**Completion routines (extended I/O)**): Երբ հասնում է ներածման/արտածման գործողությունների ավարտը, համակարգը կանչում է հատուկ ավարտման պրոցեդուրա, որն իրականացվում է հոսքի ներսում:

Բոլոր ՕՉ-երի հիմնական պրոբլեմը այն գործողություններն են, որոնք պահանջում են գործարկման երկարատև ժամանակ: Օգտագործողը գործարկում է այդպիսի ծրագիր, և այն արգելափակում է ծրագրի հիմնական

հոսքն այնքան ժամանակ, մինչև չավարտվի այդ ծրագրի աշխատանքը: Արդյունքում մենք ունենում ենք օգտագործողի համար անհարմար ՕՀ և դժգոհ հաճախորդներ:

Ասիսիսթոնացումը ներկայումս համարվում է բավականին տարածված թեմա: Հիմնականում այն օգտագործվում է ցանցային ծրագրավորման համար, սակայն բացի դրանից լինում են այնպիսի դեպքեր, երբ ծրագրի աշխատունակության և ժամանակի ծախսման քչացման համար նույնպես օգտագործում են նշված մոտեցումը:

Ասիսիսթոն ներածման/արտածման գործողությունների համար **Windows**-ում նախատեսված են երեք մեթոդներ: Ամենատարածված և ամենահարմար մեթոդ է հանդիսանում այն, որը հիմնված է հոսքերի օգտագործման վրա: Յուրաքանչյուր հոսք պատասխանում է որոշակի սահմանված հաջորդականությամբ գործողությունների իրականացմանը: Այդ գործողությունները կարող են բաղկացած լինել մեկ կամ մի քանի հաջորդաբար իրականացվող, արգելափակվող ներածման/արտածման գործողությունից: Բացի դրանից, յուրաքանչյուր հոսք պետք է տնօրինի սեփական ֆայլային դեսկրիպտորով կամ կապուղով [6]:

Փոխաձայնվող ներածում/արտածումը հնարավորություն է տալիս, օգտագործելով մեկ ֆայլային դեսկրիպտոր, մեկ հոսքով իրականացնել ասիսիսթոն գործողություններ, բայց առանձին գործողության համար «հոսք–ֆայլային դեսկրիպտոր» զույգի փոխարեն պետք է ներկայացվի պատահարի դեսկրիպտորը: Այդ դեպքում պահանջվում է կազմակերպել գործարկման ավարտման սպասում, յուրաքանչյուր ներածման/արտածման գործողության համար առանձին և հետո մաքրել համակարգային ռեսուրսները կամ էլ կատարել հաջորդական գործարկման ղեկավարման համար այլ գործողություններ:

Մյուս կողմից, ընդլայնված ներածում/արտածումը ավտոմատ կանչում է ավարտման կոդը և չի պահանջում ավելորդ պատահարների օգտագործում:

Փոխաձայնվող ներածման/արտածման մեկ անվիճելի առավելություններից է հանդիսանում այն, որ այն հնարավորություն է տալիս ստեղծել ավարտման պորտեր, սակայն այդ առավելության գինը քչանում է այն պատճառով, որ ակտիվ աշխատող հոսքերի խմբի մեջ կարող են օգտագործված լինել սեմաֆորներ: Ավարտման պորտերի լրացուցիչ թերություն է հանդիսանում այն, որ դրանք չեն թույլատրում ջնջել նրանց միացված դեսկրիպտորները:

Գրականություն

1. **Харт Джонсон**, «Системное программирование в среде Windows», Москва, Санкт-Петербург, Киев, 2005г.
2. **Вирт Н.**, Алгоритмы и структуры данных, М.: Мир, 1989г.
3. **Beveridge, Ji and Wiener, Robert**, Moltithreading Applications in Win32, 1997, Addison-wesley developers press, 376 pg.
4. **Brain, Marshall and Reeves, Ron**, Win32 System Services, 2000, Prentice Hall, 720 pg
5. **Cohen, Aaron, Woodring, Mike and Petrusha, Ronald**, Win32 Multithreaded Programming, 1998, December 11, 1997 by O'Reilly Media, 724 pages, Paperback
6. **Douglas C. Schmidt**, C++ Network Programming: Mastering Complexity with ACE and Patterns, 2001, Addison-Wesley Professional, 336 pg.

Հոդվածը տպագրության է երաշխավորել խմբագրական խորհրդի անդամ, ֆ.ս.գ.թ. Գ.Հ. Սահակյանը::