**Лабораторная работа №16**

**Организация фоновых вычислений**

**Цель работы:** Научиться выносить длительные вычисления в фоновые потоки.

**Задание:** Создайте приложение, выполняющее длительные вычисления в фоновом потоке.

Сам вычислительный процесс можно сымитировать с помощью цикла, выполняющего условные N шагов вычислений, с искусственными паузами (wait(100)) между шагами.

Интерфейс пользователя должен состоять из TextView, в котором будет отображать прогресс по работе, ProgressBar-а и кнопки Button, запускающей вычисления.

По нажатию кнопки она должна блокироваться, поверх TextView отображаться ProgressBar и создаваться новый поток, выполняющий вычисления. После завершения каждого промежуточного шага должна обновляться информация в TextView, а по завершению работы – скрываться ProgressBar и разблокироваться кнопка.

Корректно обработайте изменения состояния Activity, связанные с поворотом экрана и переключением пользователя на другое приложение – фоновый поток не должен прерываться или терять связь с Activity.

Ход работы:

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

public final static int STATUS\_FINISH = 0;

public final static int STATUS\_RUNNING = 1;

private final static int BGTASK\_CODE = 1;

public final static String PARAM\_PINTENT = "pendingIntent";

public final static String PARAM\_RESULT = "result";

int percentage = 0;

TextView percentage\_tv;

Button start\_btn;

ProgressBar PB1;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

PB1 = findViewById(R.id.progressBar1);

percentage\_tv = findViewById(R.id.percentage\_tv);

start\_btn = findViewById(R.id.start\_btn);

start\_btn.setOnClickListener(this);

}

@Override

public void onClick(View view) {

percentage=0;

update\_percentage();

start\_btn.setEnabled(false);

PB1.setVisibility(View.VISIBLE);

PendingIntent pi;

Intent intent;

pi = createPendingResult(BGTASK\_CODE, new Intent(), 0);

intent = new Intent(this, BGService.class).putExtra(PARAM\_PINTENT, pi);

startService(intent);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

// Ловим сообщения о старте задач

if (resultCode == STATUS\_FINISH) {

PB1.setVisibility(View.INVISIBLE);

start\_btn.setEnabled(true);

percentage = 0;

}

percentage = data.getIntExtra(PARAM\_RESULT, 0);;

update\_percentage();

}

private void update\_percentage(){

percentage\_tv.setText(percentage+"%");

PB1.setProgress(percentage);

}

@Override

public void onSaveInstanceState(Bundle saveInstanceState){

saveInstanceState.putInt("PERCENTAGE",percentage);

saveInstanceState.putInt("PB\_VISIBILITY",PB1.getVisibility());

saveInstanceState.putBoolean("BTN\_STATUS",start\_btn.isEnabled());

super.onSaveInstanceState(saveInstanceState);

}

@Override

public void onRestoreInstanceState(Bundle saveInstanceState){

super.onRestoreInstanceState(saveInstanceState);

percentage= saveInstanceState.getInt("PERCENTAGE");

start\_btn.setEnabled(saveInstanceState.getBoolean("BTN\_STATUS"));

PB1.setVisibility(saveInstanceState.getInt("PB\_VISIBILITY"));

update\_percentage();

}

}

public class BGService extends Service {

final String LOG\_TAG = "myLogs";

final int time = 1000;

ExecutorService es;

public void onCreate() {

super.onCreate();

Log.d(LOG\_TAG, "MyService onCreate");

es = Executors.newFixedThreadPool(1);

}

public void onDestroy() {

Log.d(LOG\_TAG, "MyService onDestroy");

super.onDestroy();

}

public int onStartCommand(Intent intent, int flags, int startId) {

Log.d(LOG\_TAG, "MyService onStartCommand");

PendingIntent pi = intent.getParcelableExtra(MainActivity.PARAM\_PINTENT);

MyRun mr = new MyRun(time, startId, pi);

es.execute(mr);

return super.onStartCommand(intent, flags, startId);

}

public IBinder onBind(Intent arg0) {

return null;

}

class MyRun implements Runnable {

int time;

int startId;

PendingIntent pi;

int percentage;

public MyRun(int time, int startId, PendingIntent pi) {

this.time = time;

this.startId = startId;

this.pi = pi;

}

public void run() {

try {

Intent intent = new Intent().putExtra(MainActivity.PARAM\_RESULT, percentage);

while (percentage < 100) {

percentage += 2;

intent.putExtra(MainActivity.PARAM\_RESULT, percentage);

pi.send(BGService.this, MainActivity.STATUS\_RUNNING, intent);

Thread.sleep(1000);

}

percentage = 0;

intent.putExtra(MainActivity.PARAM\_RESULT, percentage);

pi.send(BGService.this, MainActivity.STATUS\_FINISH, intent);

} catch (Exception e) {

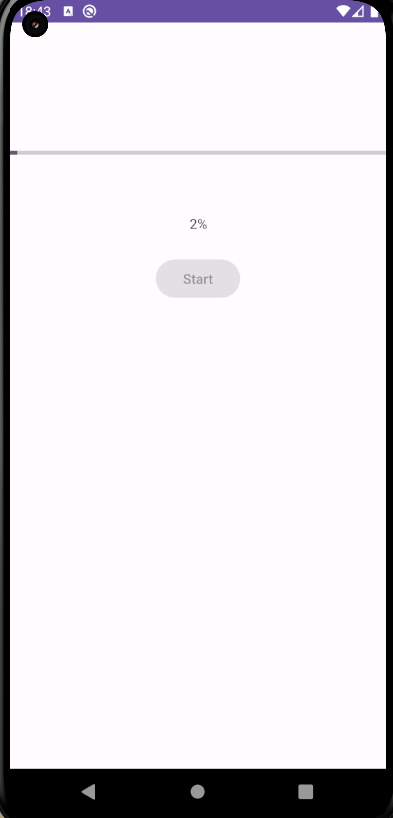
throw new RuntimeException(e);

}

}

}

}

****

**Вывод:** в ходе работы мы научились выносить длительные вычисления в фоновые потоки