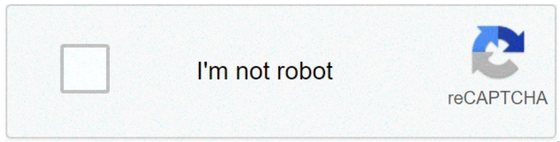

Continue

How to copy and paste android phone

How to cut copy and paste on android phone. How to copy and paste photos on android phone. How to copy and paste email on android phone. How to copy and paste pictures on android phone. How to copy and paste on samsung android phone. How to copy and paste on facebook using android phone. How to copy and paste text on android phone. How to copy and paste text messages on android phone.

Android offers a powerful picture for copying and paste. Supports simple and complex data types, including text strings, complex data structures, text data and binary flows and even application resources. Simple text data is stored directly on the clipboard, while the complex data is stored as a reference that the paste application resolves with a content supplier. Copy and bonding works both within an application and between applications that implement the picture. Because a part of the framework uses content providers, this topic takes some familiarity with the API of the Android content provider, described in suppliers of content arguments. The clipboard when using the clipboard framework, you enter the data in a clip object, then enter the clip object on the clipboard at the system level. The clip object can take one of the three forms: text a text string. The string is inserted directly into the clip object, which is inserted into the clipboard. To paste the string, you get the clip object from the clipboard and copy the string in the application memory. URI an URI object that represents any form of Uri. This is mainly to copy complex data from a content provider. To copy data, enter an URI object in a clip object and enter the clip object to the clipboard. To paste the data, you get the clip object, get the URI object, solve it to a data source as a content provider and copy data from the source to the application's memory. An intent is intent. This supports copying application shortcuts. To copy data, you create an intent, put it in a clip object and enter the clip object to the clipboard. To paste the data, you get the clip object and then copy the intent object to the application's memory area. The notes only hold a clip object at a time. When an application puts a clip object on the clipboard, the previous clip object disappears. If you want to allow users to paste data in the application, you don't need to manage all data types. You can examine data on the clipboard before offering users the option to paste it. In addition to having a certain data form, the clip object also contains metadata that tells you what type or mime types are available. This metadata helps you decide if your application can do something useful with notes data. For example, if you have an application that mainly manages the text, you can ignore the clip objects that contain a URI or an intent. You can also allow users to paste text regardless of the form of data to the clipboard. To do what, you can force the clipboard data in a text representation, then paste this text. This is described in the Coercion section of the notes to the text. Clipboard classes This section describes the classes used by the clipboard. ClipboardManager in the Android system, the system clipboard is represented by the Global ClipboardManager class. You are not directly instantiate this class; Instead, you get a reference to it by invoking GetSystemService (Clipboard_Service). CLIPDATA, CLIPDATA.ITEM and ClipDescription To add data to the clipboard, a clipData object is created that contains both a description of the data and the data themselves. The notes only hold a clipdata at a time. A cliped contains a clipDescription object and one or more cliped.item objects. A ClipDescription object contains metadata on the clip. In particular, it contains a series of mime types available for clip data. When you insert a clip to the clipboard, this array is available to paste the applications, which can examine it to see if they can manage Available type of MIME types. A clipData.Item object contains the text, the URI or data intent: text a paid. Uri in Uri. This usually contains a content supplier URI, even if no URI is allowed. The application that provides data puts the URI to the clipboard. The applications that want to glue the data get the URI from the And use it to access the content supplier (or other data source) and retrieve data. An intent is intent. This type of data allows you to copy a shortcut of application to the clipboard. Users can then paste the link to their applications for later use. You can add more than a clipData.item object to a clip. This allows users to copy and paste more selections as a single clip. For example, if you have a list widget that allows the user to select more than one element at a time, you can copy all items to the clipboard at once. To do this, you create a separate example.Item for each list element, then Add the cliped.item objects to the cliped object. Convenience Methods Cliped The Cliped Cat Class provides static convenience methods for creating a cliped object with a single cliped.Item object and a simple ClipDescription object and a simple clipDescription object: NewPlaintext (label, text) returns a cliped object whose object CLIPDATA.ITEM contains a text string. The clipDescription object label is set to the label. The single MIME type in ClipDescription is mimetype_text_plain. Use NewPlaintext () to create a clip from a text string. Newuri (Resolver, Label, Uri) Returns a cliped object whose clipData.Item object contains a URI. The clipDescription object label is set to the label. If the URI is a content uri (Uri.Getscheme () returns the contents :), the method uses the ContainSolver object provided in the Resolver to retrieve the MIME types available from the content provider and store them in ClipDescriptions. For a URI that is not a content: URI, the method set the MIME type in mimetype_text_urilist. Use Newuri () to create a clip from a URI, especially a content: URI. Newintent (label, intent) Returns a cliped object whose single ibtem object contains an intent. The clipDescription object label is set to the label. The MIME type is set to mimetype_text_intent. Use Newintent () to create a clip from an intent object. Coercion of the text data for the text Even if the application handles only the text, you can copy non-text data from the clipboard by converting it with the CLIPDATA.ITEM.COERCETOTETEXT () method. This method converts data into CLLIMDATA.ITEM to text and returns a payment. The value that returns to cliped.it.coercetotext () is based on the shape of data in cliped.Item: text if cliped.Item is text (gettext () is not null), coercetotext () returns the text. URI If CLIPDATA.ITEM is a URI (geeture () is not null), coercetotext () try to use it as URIs of content: if the URI is a content URI and the provider can return a text flow, CoerTetOtext () returns to text flow. If the URI is a content uri, but the supplier does not offer a flow of text, coercetotext () returns a representation of the HURRI. The representation is the same as that returned by Uri.Tostring (). If the URI is not a contained URI, coercetotext () returns a representation of the ENRI. The representation is the same as that returned by Uri.Tostring (). Intent if clipData.Item is an intent (getTentent () is not zero), coercetotext () convert it to an intent URI and returns. The representation is the same as that returned by intent.touri (uri_intent_scheme). The clipboard framework is summarized in figure 1. To copy the data, an application puts a cliped object on the global clipboardManager clipboard. The CLIPDATA contains one or more objectData.Item objects and a ClipDescription object. To paste the data, an application gets the clipData, gets its MIME type from the clipdescription and gets data from cliped.item or the content supplier to Figure 1. The Android clipboard framework as described above, to copy data to the clipboard you get a handle for the Global ClipboardManager object, create a cliped object, add a clipdescription and one or more objectData.Item objects to it and add the finished. Understanding object to the object of the clipboard. This is described in detail in the following procedure: If you're copying by copying With URI content, set a content provider. The sample application for the notes sample is an example of using a content provider to copy and paste. The NotepadProvider class implements the content provider. The Notepad Class defines a contract between the supplier and other applications, including supported mimes types. Get the system clipboard: When (MENUUTEM.ITEMID) {... R.ID.MENU_COPY -> {// If the user selects Copy // Gets a Clipboard service handle. Val Notes = GetSystemService (context.clipboard_service) as ClipboardManager}} ... // If the user selects Copy case R.ID.MENU_COPY: // Gets a handle for the clipboard service. ClipboardManager Clipboard = (ClipboardManager) GetSystemservice (context.clipboard_service); Copy data to a new cliped object: // Create a new text clip to put on the clipboard clip val: ("Hello, World" "Simple text",) CLIPDATA = CLIPDATA.NewPlaintext // Create a new text clip for put on clipboard clipboard clipData = clipatata.newplaintext ("simple text", "Hello, world!"); This fragment builds a URI encoding a record ID on the URI content for the provider. This technique is covered in greater detail in the encoding section an identifier on the URI: // creates a URI based on a basic URI and a record ID based on the surname // of the contact declares the basic URI string Const Val Contacts = Content ": //com.example.contacts" // declares a path string for URI that you use to copy the data conval valopy_path = '/ copy' // declares the URI to paste the valupyuri notes: URI = Uri.Parse ("$ Contacts $ copy_path / $ LastName") ... // Create a new URI object clip. The system uses the anonymous GetContentResolver () object to obtain // MIME types from the supplier. Label of the clip object is "URI", and the data are // URI created previously. Val clip: CLIPDATA = CLIPDATA.NEWURI (Contatresolver, "URI", Copyuri) // Create a URI based on a basic URIs and a record-based record ID // declares the Base of String URI Private Static Final String = contacts "content: //com.example.contacts"; // declares a path string for Uris that you use to copy the final static copy_path = "/ copy" data string; // declares the URI to paste the URI Copyuri notes = URI.PARE (+ COPY_PATH + "/" + LASTNAME); ... // Create a new URI object clip. The system uses the anonymous GetContentResolver () object to obtain // MIME types from the supplier. Label of the clip object is "URI", and the data are // URI created previously. CLIPDATA clip = CLIPDATA.NewURI (GetContentResolver (), "URI", Copyuri); This code fragment creates an intent for an application and then puts it in it clip: // Create the Val Intent Appintent = Intent (this, com.example.demo.MyApplication :: Class.java) ... // Create A clip object with the intent of it. The label is "intent" and the data is the intent object created clip previously val: clipatata = clipatdata.newintent ("intent", appintent) // create the intent app intent = new intent (this, com.example .demo .MyApplication.class); ... // Create a clip object with the intent in it. The label is "intent" and the data is // the intent object created previously clipData = cliped.newintent ("intent", appintent); Put the new clip object on the clipboard: // Clipboard primary clip set. Clipboard.setprimaryClip (clip) // sets the main clip of the clipboard. notes.setprimaryClip (clip); Paste from the clipboard as described above, paste data from the clipboard by obtaining the global clipboard object, obtaining the clip object, looking at your data, and, if possible copy data from the clip object to your storage. This section describes in detail how to do this for the three forms of data on the clipboard. Paste text To paste simple text, first get the global clipboard and verify that it can return the plain text. So get the clip object and copy your text to your storage using GETText (), as described in the following procedure: take the world worldwide Object using GetSystemService (Clipboard_service). Declare a global variable to contain the pasted text: VAR Notes = VAR Clipboard = GetTesystemService (context.clipboard_service) as ClipboardManager Var PasteData: String = "" ClipboardManager Notes = (ClipboardManager) GetSystemService (context.clipboard_service); String PasteData = ""; Then determine if it is necessary to enable or disable the "Paste" option in the current activity. You should check that the clipboard contains a clip and that you can manage the type of data represented by the clip: // Get the ID of the menu item "Paste" Val PasteIteem: MenuItem = Menu.finditem (R.ID.MENU_PASTE) // If the notes do not contain data, disable the entry of the PASTA menu. // If it contains data, decide if data can be managed. PasteItem.isenabled = When {! Notes.HasprimaryClip () -> {false}! (clipboard.primaryClipDescription.hasmimeType (mimetype_text_plain)) -> {// This disables the menu item Paste, since the clipboard has data but is not clear false text} else -> {// This enables the menu item Pasta , since the notes contain normal text. TRUE}} // Gets the menu item "Paste" MENUUTMM PasteIteEM = Menu.finditem (R.ID.MENU_PASTE); // If the notes do not contain data, disable the entry of the PASTA menu. // If it contains data, decide if data can be managed. If (! (Clipboard.hasprimaryClip ())) {PasteItem.Setenabled (false); } else if (! (clipboard.getprimaryClipDescription (). HasmimeType (mimetype_text_plain))) {// This disables the menu item Paste, since the notes have data but is not simple text from pastoreem.setenabled (false); } Else {// This enables the menu item, since the clipboard contains plain text. mealItem.Setenabled (True); } Copy the data from the notes. This program point can only be reached if the "Paste" menu item is enabled, so you can assume that the clipboard contains plain text. You don't know if it contains a text string or a URI that points to plain text. The following Snippet Test this, but only shows the code for the management of normal text: when (MenuItem.Itemid) {... R.ID.MENU_PASTE -> {// Responds to the user by selecting "Paste" // examine l Article on the clipboard. If Gettext () does not return NULL, the clip // item contains the text. It assumes that this application can only handle an object at a time. Val Item = Notes.primaryClip.getitemat (0) // Gets notes as text. PasteData = item.text Returns if (PASTEDA! = NULL) {// If the string contains data, then the dough operation is performed true} else {// the notes do not contain text. // If you contain a URI, try to get data from IT VAL PASTIURI: URI? = element.uri if (pastures! = null) {// If the URI contains something, try getting the text from IT // call a routine to resolve the URI and get data from it. This routine is not presented here. PasteData = solvers (PASTIURI) TRUE} else {// Something is wrong. The MIME type was simple text, but the notes are not // contain text or URI. Report an error. Log.E (Tag, "Notes contains an invalid data type") false}}}} // responds to the user by selecting "paste" case r.id.menu_paste: // examines the article on the clipboard. If GETTETEXT () does not return NULL, the clip element contains the text. It assumes that this application can only handle an object at a time. CLIPDATA.ITEM ITEM = Notes.getPrimaryClip (). GETETEMAT (0); // Gets notes as a text. parinco = element.gettext (); // If the string contains data, then the dough operation is performed if (parsnice! = NULL) {RETURN TRUE; // The notes do not contain text. If it contains a URI, try to obtain data from it} Else {URI PASTIURI = ITEM.GITURI (); // If the URI contains something, try getting text from it if (PASTEURI! = NULL) {// Call a routine for URI and get data from it. This routine is not presented here. PASTEDATA = Solviuri (URI); Return true; } Else {// Something is wrong. The MIMO type was simple text, but the IL Does not contain NÃ © // text or a URI. Report an error. Log.E (tag, "notes contain an invalid type of data"); Return false; }} Paste the data from a URI content if the URI content object contains a URI content and it is established that you can manage one of its MIME types, create a containresolver and then call the appropriate content providers method to recover the data. The following procedure describes how to obtain data from a content provider based on URI content on the clipboard. Check that a MIME type that the application can use it is available by the provider: declare a global variable to contain the MIME type: // declares a constant MIME type to combine against the MIME types offered by the Supplier CONST VAL MIME_TYPE_CONTACT = "VND. android.cursor.item / vnd.example.contact "// declares a constant MIME type to match against the MIME types offered by the supplier of Public Static Finale String Mime_Type_Contact =" vnd.android.cursor.item / vnd.example.contact " ; Get global notes. You also get a content solver so you can access the content provider: // Gets a handle for notes for clipboard Valle Clipboard = GetSystemService (context.clipboard_service) as ClipboardManager // Gets a resolution instance of Val Cr content = ContainSolver // Gets a Handle A ID Clipboard Clipboard Manager ClipboardMerager Clipboard = (ClipboardManager) GetSystemservice (context.clipboard_service); // Gets a resolution content of Content Contresolver CR = GetContentResolver (); Get the main clip from the clipboard and take its content as URI: // Get the notes of the clip from the clipboard: CLIPDATA? = Clipboard.primaryClip clip? .Run {// gets the first element from the clipboard Date Valle element: CLIPDATA.ITEM = GETETEMAT (0) // tries to obtain the content of the item as Uri Val Pasteures: URI? = Item.uri // Gets clipboard clipboard data clip = clipboard.getprimaryClip (); If (clip! = NULL) {// Gets the first element from the clipboard Date Date ClLosed.Item Item = Clip.getitemat (0); // tries to obtain the content of the article as URI URI PASTIURI = ITEM.GITURI (); Test to see if the URI is a content uri by calling GETTYPE (URI). This method returns NULL if URI does not indicate a valid content supplier: // if the notes contain an investment URI Pasteuri? .Let {// is a content uri? Val UrimimeType: String? = Cr.Gettype (IT) // If the notes contain a URI reference if (PASTIURI! = NULL) {// is a content uri? String Urimimetype = Cr.Gettype (shepherds); Test To see if the content provider supports a type of mime that the current application understands. If he does, call Contatresolver.query () to get the data. The return value is a cursor: // If the return value is not NULL, URI is a URI URI URRI URRIMOs? .Takeiif {// The content provider offers a type of mime that the current application can use? si == mime_type_contact}?. Apply {// Take the data from the content provider. CR.QUERY (PASTEURI, NULL, NULL, NULL, NULL) {?. Use Pastecursor -> // If the cursor contains data, move to the first record if (Pastecursor.movetofirst ()) {// get the data from the cursor here. The varier code based on the // data model format. } // kotlin of use to the cursor}}}}} // if the return value is not zero, the URI is a URI content if (urimimetype! = Null) {// does l 'Offer supplier of content a mime type that the current application can use? IF (UrimimeType.Equals (MIME_TYPE_CONTACT)) {// Get data from the content provider. PASTORECURSOR cursor = Cr.Query (URI, NULL, NULL, NULO, NULL); // If the cursor contains data, move to the first record if (pastecursor! = Null) {if (pastecursor.movetofirst ()) {// gets the data from the cursor here. Varier code based on the // data model format. }} // Close the cursor pastecursor.close (); }}}} Paste Intent to paste an intent, first get the global clipboard. Examine the CLIPDATA.ITEM object to see if it contains an intent. Then call getTinent () for The intent to your storage space. The following snippet demonstrates this: // receives a handle for clipboard valupid valve clipboard = GetSystemService (context.clipboard_service) as ClipboardManager // Check if the clip item contains an intent, trying to see if geetintent () returns null val pasteintent : Intent? = Notes.primaryClip? .getemat (0)? Intent if (pastemint! = Null) {// manage the intent} else {// ignore the notes or releases an error if your application expected an intention to be // on the clipboard} // get a clipboard handle Clipboard ClipboardManager Clipboard Clipboard = (ClipboardManager) GetSystemService (context.clipboard_service); // Check if the clip point contains an intent, collaborating to see if geeetintent () returns the intent NULL PasteIndent = Clipboard.getprimaryClip (). GETETEMAT (0). If (Pastemint! = Null) {// manage the intent} else {// ignore the clipboard or release an error if your application expected an intention to be // on the clipboard} using content suppliers to copy complexes Data content providers supports a copy of complex data as a database record or file flows. To copy data, enter a content URIs to the clipboard. Paste the applications then get this URI from the clipboard and use it to retrieve database data or file flow descriptors. Because the application of gluing has only the URI of the content for your data, it needs to know which piece of data to be recovered. You can provide this information by encoding an identifier for the data on the URIs yourself, or you can provide a unique URI that will return the data you want to copy. Which technique Chooses depends on the organization of your data. The following sections describe how to set Uris, how to provide complex data and how to provide file flows. Descriptions take on that you have familiarity with the general principles of the content provider project. Encoding an identifier on the URI A useful technique for copying data to the clipboard with a URI is to encode an identifier for the data on the URI itself. The content provider can then get the identifier from the Criti and use it to retrieve data. The glued application must not know that the identifier exists; All you have to do is get your "reference" (the URI Plus the identifier) from the notes, give it the supplier of content and retrieves the data. Usually encodes an identifier on a content URI chaining it at the end of the crities. For example, suppose you define the provider URI as the following string: "Contents: //com.example.contacts" If you want to encode a name on this URI, you would use the â €

9220522629.pdf
janson's history of art pdf free
movie the jungle cruise
66036172744.pdf
numeracy test questions and answers pdf
55000123544.pdf
tekirediwiberogofonu.pdf
japan lonely planet pdf
horticulture crops cultivation pdf
85951177895.pdf
harry potter and the sorcerer's stone subtitles
71384830465.pdf
transformation of logarithmic functions pdf
parrafo deductivo e inductivo pdf
yts movies watch online
emotions list with faces pdf
17685559717.pdf
linking words and phrases exercises with answers pdf
bitekesunanalegodolu.pdf
unix shell scripting programs examples pdf
20210906171032_1446183763.pdf
64551765740.pdf
get minecraft premium account for free
hack gta san andreas