NAME

moused - pass mouse data to the console driver

SYNOPSIS

moused [**-DPRacdfs**] [**-I** *file*] [**-F** *rate*] [**-r** *resolution*] [**-S** *baudrate*] [**-VH** [**-U** *distance* **-L** *distance*]] [**-A** *exp*[,*offset*]] [**-a** *X*[,*Y*]] [**-C** *threshold*] [**-m** *N=M*] [**-w** *N*] [**-z** *target*] [**-t** *mousetype*] [**-l** *level*] [**-3** [**-E** *timeout*]] [**-T** *distance*[,*time*[,*after*]]] **-p** *port*

moused [**-Pd**] **-p** *port* **-i** *info*

DESCRIPTION

The **moused** utility and the console driver work together to support mouse operation in the text console and user programs. They virtualize the mouse and provide user programs with mouse data in the standard format (see sysmouse(4)).

The mouse daemon listens to the specified port for mouse data, interprets and then passes it via ioctls to the console driver. The mouse daemon reports translation movement, button press/release events and movement of the roller or the wheel if available. The roller/wheel movement is reported as "Z" axis movement.

The console driver will display the mouse pointer on the screen and provide cut and paste functions if the mouse pointer is enabled in the virtual console via vidcontrol(1). If sysmouse(4) is opened by the user program, the console driver also passes the mouse data to the device so that the user program will see it.

If the mouse daemon receives the signal SIGHUP, it will reopen the mouse port and reinitialize itself. Useful if the mouse is attached/detached while the system is suspended.

If the mouse daemon receives the signal SIGUSR1, it will stop passing mouse events. Sending the signal SIGUSR1 again will resume passing mouse events. Useful if your typing on a laptop is interrupted by accidentally touching the mouse pad.

The following options are available:

-3 Emulate the third (middle) button for 2-button mice. It is emulated by pressing the left and right physical buttons simultaneously.

-C *threshold*
Set double click speed as the maximum interval in msec between button clicks. Without this option, the default value of 500 msec will be assumed. This option will have effect only on the

cut and paste operations in the text mode console. The user program which is reading mouse data via sysmouse(4) will not be affected.

-D Lower DTR on the serial port. This option is valid only if *mousesystems* is selected as the protocol type. The DTR line may need to be dropped for a 3-button mouse to operate in the *mousesystems* mode.

-E *timeout*
When the third button emulation is enabled (see above), the **moused** utility waits *timeout* msec at most before deciding whether two buttons are being pressed simultaneously. The default timeout is 100 msec.

-F *rate*
Set the report rate (reports/sec) of the device if supported.

-L *distance*
When "Virtual Scrolling" is enabled, the **-L** option can be used to set the *distance* (in pixels) that the mouse must move before a scroll event is generated. This effectively controls the scrolling speed. The default *distance* is 2 pixels.

-H Enable "Horizontal Virtual Scrolling". With this option set, holding the middle mouse button down will cause motion to be interpreted as horizontal scrolling. Use the **-U** option to set the distance the mouse must move before the scrolling mode is activated and the **-L** option to set the scrolling speed. This option may be used with or without the **-V** option.

-I *file* Write the process id of the **moused** utility in the specified file. Without this option, the process id will be stored in */var/run/moused.pid*.

-P Do not start the Plug and Play COM device enumeration procedure when identifying the serial mouse. If this option is given together with the **-i** option, the **moused** utility will not be able to print useful information for the serial mouse.

-R Lower RTS on the serial port. This option is valid only if *mousesystems* is selected as the protocol type by the **-t** option below. It is often used with the **-D** option above. Both RTS and DTR lines may need to be dropped for a 3-button mouse to operate in the *mousesystems* mode.

-S *baudrate*
Select the baudrate for the serial port (1200 to 9600). Not all serial mice support this option.

-T *distance*[,*time*[,*after*]]

Terminate drift. Use this option if mouse pointer slowly wanders when mouse is not moved. Movements up to *distance* (for example 4) pixels (X+Y) in *time* msec (default 500) are ignored, except during *after* msec (default 4000) since last real mouse movement.

-V Enable "Virtual Scrolling". With this option set, holding the middle mouse button down will cause motion to be interpreted as scrolling. Use the **-U** option to set the distance the mouse must move before the scrolling mode is activated and the **-L** option to set the scrolling speed.

-U *distance*

When "Virtual Scrolling" is enabled, the **-U** option can be used to set the *distance* (in pixels) that the mouse must move before the scrolling mode is activated. The default *distance* is 3 pixels.

-A *exp*[,*offset*]

Apply exponential (dynamic) acceleration to mouse movements: the faster you move the mouse, the more it will be accelerated. That means that small mouse movements are not accelerated, so they are still very accurate, while a faster movement will drive the pointer quickly across the screen.

The *exp* value specifies the exponent, which is basically the amount of acceleration. Useful values are in the range 1.1 to 2.0, but it depends on your mouse hardware and your personal preference. A value of 1.0 means no exponential acceleration. A value of 2.0 means squared acceleration (i.e. if you move the mouse twice as fast, the pointer will move four times as fast on the screen). Values beyond 2.0 are possible but not recommended. A good value to start is probably 1.5.

The optional *offset* value specifies the distance at which the acceleration begins. The default is 1.0, which means that the acceleration is applied to movements larger than one unit. If you specify a larger value, it takes more speed for the acceleration to kick in, i.e. the speed range for small and accurate movements is wider. Usually the default should be sufficient, but if you're not satisfied with the behaviour, try a value of 2.0.

Note that the **-A** option interacts badly with the X server's own acceleration, which doesn't work very well anyway. Therefore it is recommended to switch it off if necessary: "xset m 1".

-a *X*[,*Y*]

Accelerate or decelerate the mouse input. This is a linear acceleration only. Values less than 1.0 slow down movement, values greater than 1.0 speed it up. Specifying only one value sets the acceleration for both axes.

You can use the **-a** and **-A** options at the same time to have the combined effect of linear and exponential acceleration.

-c Some mice report middle button down events as if the left and right buttons are being pressed. This option handles this.

-d Enable debugging messages.

-f Do not become a daemon and instead run as a foreground process. Useful for testing and debugging.

-i *info* Print specified information and quit. Available pieces of information are:

port	Port (device file) name, i.e. */dev/cuau0*, and */dev/psm0*.
if	Interface type: serial, bus, inport or ps/2.
type	Protocol type. It is one of the types listed under the **-t** option below or *sysmouse* if the driver supports the *sysmouse* data format standard.
model	Mouse model. The **moused** utility may not always be able to identify the model.
all	All of the above items. Print port, interface, type and model in this order in one line.

If the **moused** utility cannot determine the requested information, it prints "unknown" or "generic".

-l *level*
Specifies at which level **moused** should operate the mouse driver. Refer to *Operation Levels* in psm(4) for more information on this.

-m *N=M*
Assign the physical button *M* to the logical button *N*. You may specify as many instances of this option as you like. More than one physical button may be assigned to a logical button at the same time. In this case the logical button will be down, if either of the assigned physical buttons is held down. Do not put space around ‘=’.

-p *port*
Use *port* to communicate with the mouse.

-r *resolution*
Set the resolution of the device; in Dots Per Inch, or *low*, *medium-low*, *medium-high* or *high*. This option may not be supported by all the device.

-s Select a baudrate of 9600 for the serial line. Not all serial mice support this option.

-t *type*

Specify the protocol type of the mouse attached to the port. You may explicitly specify a type listed below, or use *auto* to let the **moused** utility automatically select an appropriate protocol for the given mouse. If you entirely omit this option in the command line, **-t** *auto* is assumed. Under normal circumstances, you need to use this option only if the **moused** utility is not able to detect the protocol automatically (see *Configuring Mouse Daemon*).

Note that if a protocol type is specified with this option, the **-P** option above is implied and Plug and Play COM device enumeration procedure will be disabled.

Also note that if your mouse is attached to the PS/2 mouse port, you should always choose *auto* or *ps/2*, regardless of the brand and model of the mouse. Likewise, if your mouse is attached to the bus mouse port, choose *auto* or *busmouse*. Serial mouse protocols will not work with these mice.

For the USB mouse, the protocol must be *auto*. No other protocol will work with the USB mouse.

Valid types for this option are listed below.

For the serial mouse:

microsoft	Microsoft serial mouse protocol. Most 2-button serial mice use this protocol.
intellimouse	Microsoft IntelliMouse protocol. Genius NetMouse, ASCII Mie Mouse, Logitech MouseMan+ and FirstMouse+ use this protocol too. Other mice with a roller/wheel may be compatible with this protocol.
mousesystems	MouseSystems 5-byte protocol. 3-button mice may use this protocol.
mmseries	MM Series mouse protocol.
logitech	Logitech mouse protocol. Note that this is for old Logitech models. *mouseman* or *intellimouse* should be specified for newer models.
mouseman	Logitech MouseMan and TrackMan protocol. Some 3-button mice may be compatible with this protocol. Note that MouseMan+ and FirstMouse+ use *intellimouse* protocol rather than this one.
glidepoint	ALPS GlidePoint protocol.
thinkingmouse	Kensington ThinkingMouse protocol.
mmhitab	Hitachi tablet protocol.
x10mouseremote	X10 MouseRemote.
kidspad	Genius Kidspad and Easypad protocol.
versapad	Interlink VersaPad protocol.
gtco_digipad	GTCO Digipad protocol.

For the bus and InPort mouse:

busmouse This is the only protocol type available for the bus and InPort mouse and should be specified for any bus mice and InPort mice, regardless of the brand.

For the PS/2 mouse:

ps/2 This is the only protocol type available for the PS/2 mouse and should be specified for any PS/2 mice, regardless of the brand.

For the USB mouse, *auto* is the only protocol type available for the USB mouse and should be specified for any USB mice, regardless of the brand.

-w *N* Make the physical button *N* act as the wheel mode button. While this button is pressed, X and Y axis movement is reported to be zero and the Y axis movement is mapped to Z axis. You may further map the Z axis movement to virtual buttons by the **-z** option below.

-z *target*
Map Z axis (roller/wheel) movement to another axis or to virtual buttons. Valid *target* maybe:

x
y X or Y axis movement will be reported when the Z axis movement is detected.

N Report down events for the virtual buttons *N* and *N+1* respectively when negative and positive Z axis movement is detected. There do not need to be physical buttons *N* and *N+1*. Note that mapping to logical buttons is carried out after mapping from the Z axis movement to the virtual buttons is done.

N1 N2
Report down events for the virtual buttons *N1* and *N2* respectively when negative and positive Z axis movement is detected.

N1 N2 N3 N4
This is useful for the mouse with two wheels of which the second wheel is used to generate horizontal scroll action, and for the mouse which has a knob or a stick which can detect the horizontal force applied by the user.

The motion of the second wheel will be mapped to the buttons *N3*, for the negative direction, and *N4*, for the positive direction. If the buttons *N3* and *N4* actually exist in this mouse, their actions will not be detected.

Note that horizontal movement or second roller/wheel movement may not always be detected, because there appears to be no accepted standard as to how it is encoded.

Note also that some mice think left is the negative horizontal direction; others may think otherwise. Moreover, there are some mice whose two wheels are both mounted vertically, and the direction of the second vertical wheel does not match the first one.

Configuring Mouse Daemon

The first thing you need to know is the interface type of the mouse you are going to use. It can be determined by looking at the connector of the mouse. The serial mouse has a D-Sub female 9- or 25-pin connector. The bus and InPort mice have either a D-Sub male 9-pin connector or a round DIN 9-pin connector. The PS/2 mouse is equipped with a small, round DIN 6-pin connector. Some mice come with adapters with which the connector can be converted to another. If you are to use such an adapter, remember the connector at the very end of the mouse/adapter pair is what matters. The USB mouse has a flat rectangular connector.

The next thing to decide is a port to use for the given interface. The PS/2 mouse is always at */dev/psm0*. There may be more than one serial port to which the serial mouse can be attached. Many people often assign the first, built-in serial port */dev/cuau0* to the mouse. You can attach multiple USB mice to your system or to your USB hub. They are accessible as */dev/ums0*, */dev/ums1*, and so on.

You may want to create a symbolic link */dev/mouse* pointing to the real port to which the mouse is connected, so that you can easily distinguish which is your "mouse" port later.

The next step is to guess the appropriate protocol type for the mouse. The **moused** utility may be able to automatically determine the protocol type. Run the **moused** utility with the **-i** option and see what it says. If the command can identify the protocol type, no further investigation is necessary on your part. You may start the daemon without explicitly specifying a protocol type (see *EXAMPLES*).

The command may print *sysmouse* if the mouse driver supports this protocol type.

Note that the type and model printed by the **-i** option do not necessarily match the product name of the pointing device in question, but they may give the name of the device with which it is compatible.

If the **-i** option yields nothing, you need to specify a protocol type to the **moused** utility by the **-t** option. You have to make a guess and try. There is rule of thumb:

1. The bus and InPort mice always use *busmouse* protocol regardless of the brand of the mouse.
2. The *ps/2* protocol should always be specified for the PS/2 mouse regardless of the brand of the mouse.
3. You must specify the *auto* protocol for the USB mouse.
4. Most 2-button serial mice support the *microsoft* protocol.
5. 3-button serial mice may work with the *mousesystems* protocol. If it does not, it may work with the *microsoft* protocol although the third (middle) button will not function. 3-button serial mice may also work with the *mouseman* protocol under which the third button may function as expected.

6. 3-button serial mice may have a small switch to choose between "MS" and "PC", or "2" and "3". "MS" or "2" usually mean the *microsoft* protocol. "PC" or "3" will choose the *mousesystems* protocol.
7. If the mouse has a roller or a wheel, it may be compatible with the *intellimouse* protocol.

To test if the selected protocol type is correct for the given mouse, enable the mouse pointer in the current virtual console,

```
vidcontrol -m on
```

start the mouse daemon in the foreground mode,

```
moused -f -p <selected_port> -t <selected_protocol>
```

and see if the mouse pointer travels correctly according to the mouse movement. Then try cut & paste features by clicking the left, right and middle buttons. Type ^C to stop the command.

Multiple Mice

As many instances of the mouse daemon as the number of mice attached to the system may be run simultaneously; one instance for each mouse. This is useful if the user wants to use the built-in PS/2 pointing device of a laptop computer while on the road, but wants to use a serial mouse when s/he attaches the system to the docking station in the office. Run two mouse daemons and tell the application program (such as the X Window System) to use sysmouse(4), then the application program will always see mouse data from either mouse. When the serial mouse is not attached, the corresponding mouse daemon will not detect any movement or button state change and the application program will only see mouse data coming from the daemon for the PS/2 mouse. In contrast when both mice are attached and both of them are moved at the same time in this configuration, the mouse pointer will travel across the screen just as if movement of the mice is combined all together.

FILES

/dev/consolectl	device to control the console
/dev/psm%d	PS/2 mouse driver
/dev/sysmouse	virtualized mouse driver
/dev/ttyv%d	virtual consoles
/dev/ums%d	USB mouse driver
/var/run/moused.pid	process id of the currently running **moused** utility
/var/run/MouseRemote	UNIX-domain stream socket for X10 MouseRemote events

EXAMPLES

```
moused -p /dev/cuau0 -i type
```

Let the **moused** utility determine the protocol type of the mouse at the serial port */dev/cuau0*. If successful, the command will print the type, otherwise it will say "unknown".

```
moused -p /dev/cuau0
vidcontrol -m on
```

If the **moused** utility is able to identify the protocol type of the mouse at the specified port automatically, you can start the daemon without the **-t** option and enable the mouse pointer in the text console as above.

```
moused -p /dev/mouse -t microsoft
vidcontrol -m on
```

Start the mouse daemon on the serial port */dev/mouse*. The protocol type *microsoft* is explicitly specified by the **-t** option.

```
moused -p /dev/mouse -m 1=3 -m 3=1
```

Assign the physical button 3 (right button) to the logical button 1 (logical left) and the physical button 1 (left) to the logical button 3 (logical right). This will effectively swap the left and right buttons.

```
moused -p /dev/mouse -t intellimouse -z 4
```

Report negative Z axis movement (i.e., mouse wheel) as the button 4 pressed and positive Z axis movement (i.e., mouse wheel) as the button 5 pressed.

If you add

```
ALL ALL = NOPASSWD: /usr/bin/killall -USR1 moused
```

to your */usr/local/etc/sudoers* file, and bind

```
killall -USR1 moused
```

to a key in your window manager, you can suspend mouse events on your laptop if you keep brushing over the mouse pad while typing.

SEE ALSO

kill(1), vidcontrol(1), xset(1), keyboard(4), psm(4), screen(4), sysmouse(4), ums(4)

STANDARDS

The **moused** utility partially supports "Plug and Play External COM Device Specification" in order to support PnP serial mice. However, due to various degrees of conformance to the specification by existing serial mice, it does not strictly follow the version 1.0 of the standard. Even with this less strict approach, it may not always determine an appropriate protocol type for the given serial mouse.

HISTORY

The **moused** utility first appeared in FreeBSD 2.2.

AUTHORS

The **moused** utility was written by Michael Smith <*msmith@FreeBSD.org*>. This manual page was written by Mike Pritchard <*mpp@FreeBSD.org*>. The command and manual page have since been updated by Kazutaka Yokota <*yokota@FreeBSD.org*>.

CAVEATS

Many pad devices behave as if the first (left) button were pressed if the user "taps" the surface of the pad. In contrast, some ALPS GlidePoint and Interlink VersaPad models treat the tapping action as fourth button events. Use the option "**-m** 1=4" for these models to obtain the same effect as the other pad devices.

Cut and paste functions in the virtual console assume that there are three buttons on the mouse. The logical button 1 (logical left) selects a region of text in the console and copies it to the cut buffer. The logical button 3 (logical right) extends the selected region. The logical button 2 (logical middle) pastes the selected text at the text cursor position. If the mouse has only two buttons, the middle, 'paste' button is not available. To obtain the paste function, use the **-3** option to emulate the middle button, or use the **-m** option to assign the physical right button to the logical middle button: "**-m** 2=3".